Predicting the Approximate Functional Behaviour of Physical Systems

Colin Paul Williams

Ph.D.

University of Edinburgh

1989

To Kathy

Abstract

This dissertation addresses the problem of the computer prediction of the approximate behaviour of physical systems describable by ordinary differential equations.

Previous approaches to behavioural prediction have either focused on an exact mathematical description or on a qualitative account. We advocate a middle ground: a representation more coarse than an exact mathematical solution yet more specific than a qualitative one. What is required is a mathematical expression, simpler than the exact solution, whose qualitative features mirror those of the actual solution and whose functional form captures the principal parameter relationships underlying the behaviour of the real system. We term such a representation an *approximate functional solution.*

Approximate functional solutions are superior to qualitative descriptions because they reveal specific functional relationships, restore a quantitative time scale to a process and support more sophisticated comparative analysis queries. Moreover, they can be superior to exact mathematical solutions by emphasizing comprehensibility, adequacy and practical utility over precision.

Two strategies for constructing *approximate functional solutions* are proposed. The first abstracts the original equation, predicts behaviour in the abstraction space and maps this back to the approximate functional level. Specifically, **analytic abduction** exploits qualitative simulation to predict the qualitative properties of the solution and uses this knowledge to guide the selection of a parameterized trial function which is then tuned with respect to the differential equation. In order to limit the complexity of a proposed *approximate functional solution,* and hence maintain its comprehensibility, **back-of-the-envelope** reasoning is used to simplify overly complex expressions in a magnitude extreme. If no function is recognised which matches the predicted behaviour, **segment calculus** is called upon to find a composite function built from known primitives and a set of operators. At the very least, **segment calculus** identifies a plausible structure for the form of the solution (*e.g.* that it is a composition of two unknown functions). **Equation parsing** capitalizes on this partial information to look for a set of termwise interactions which, when interpreted, expose a particular solution of the equation.

The second, and more direct, strategy for constructing an *approximate functional solution* is embodied in the **closed form approximation** technique. This extends approximation methods to equations which lack a closed form solution. This involves solving the differential equation exactly, as an infinite series, and obtaining an *approximate functional solution* by constructing a closed form function whose Taylor series is close to that of the exact solution.

The above techniques dovetail together to achieve a style of reasoning closer to that of an engineer or physicist rather than a mathematician. The key difference being to sacrifice the goal of finding the correct solution of the differential equation in favour of finding an approximation which is adequate for the purpose to which the knowledge will be put. Applications to Intelligent Tutoring and Design Support Systems are suggested.

Acknowledgements

I would never have completed this work had it not been for Don Gaubatz, Jim Scott and Brian Rees of Digital Equipment Corporation. Don arrived on the scene when my coffers were empty and spirits low. Through his efforts, and the generous support of the Advanced Microsystems Development Group of Digital Equipment Corporation, both states of affairs were reversed.

I would especially like to thank my parents for their continual support and encouragement. Few sons could be so lucky.

I also wish to thank my supervisors, Peter Ross and John Hallam, for their guidance and incisive comments throughout my time at Edinburgh.

A significant part of my learning experience has come from discussions with my contemporaries. In particular, I would like to thank Mike Cameron-Jones and Andy Robertson for acting as my "pet" engineers and bolstering my faith in the value of approximate methods; Gary Roberts for his indefatigable enthusiasm for matters computational and countless Socratic dialogues; Paul McIlvenny for keeping me aware of the inadequacies of current Artificial Intelligence; and Karen Valley for maintaining my sanity.

Finally I should also thank the staff and students of the Department of Artificial Intelligence for providing a stimulating technical environment and continuous *bonhomie*.

This dissertation was prepared on machines generously donated under the Rank Xerox University Grants Programme.

The first two years of this research were funded by a grant from the Science and Engineering Research Council.

Declaration

I declare that this dissertation has been composed by myself and describes my own work.

Colin P. Williams

Contents

List of Figures

List of Tables

List of Algorithms

List of Definitions

List of Theorems

Chapter 1

Introduction

1.1 The Problem

This thesis is concerned with the computer prediction of the approximate behaviour of physical systems describable by ordinary differential equations.

By *approximate behaviour* we mean a simple mathematical expression whose qualitative form is congruent to that of the real solution (at least over some interval) and which captures functional relationships between parameters that reflect the key aspects of the real solution. We call such a solution of the original differential equation an *approximate functional solution.*

The phrase "key aspects" captures the notion that if an increase in a parameter p in the real solution has an effect E then p should also exist in the approximate functional solution and its increase should entail E too. The complication is that the real solution will not always be explicitly available so it is necessary to consider indirect methods for checking the fidelity of the approximation.

The key feature of approximate functional solutions is that they introduce a new level of distinction in behavioural descriptions: they are more precise than purely qualitative

descriptions yet less precise than exact mathematical descriptions. Their advantage is that they combine the best features of one scheme to overcome the deficiencies of the other.

1.2 Motivation for Approximate Solutions

In *Chapter 2* we will discuss various Artificial Intelligence programs which use differential equations as the basis for reasoning about the behaviour of an associated physical system. These fall into one of the broad categories shown below.

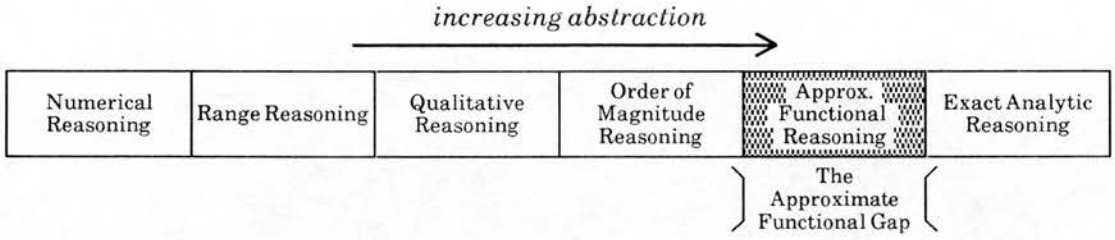


Figure 1-1. Spectrum of Reasoning Systems

Each category is distinguished by the granularity of information it employs. Moving left to right, these become a progressively more generalised concept of number.

On the far left sits numerical reasoning which is, without doubt, the most widely used technique in engineering applications today. Moving to the right, individual numbers become abstracted into ranges or qualitative values. This level of representation was developed to enable all distinct modes of behaviour to be predicted given only a partial system specification. Slightly further along, another abstraction comes into play, based on hierarchies of relative magnitudes. On the far right, exact analytic reasoning allows differential equations to be solved to produce a mathematical function which describes the system behaviour.

We claim there is a gap in this spectrum between qualitative and analytic reasoning which we designate "the approximate functional level". The distinguishing features of this level are discussed in the following sections.

1.2.1 Comprehensibility as a Goal

Approximate functional reasoning specifically aims to obtain a *comprehensible* description of behaviour. This is in contrast to the other approaches which relegate comprehensibility to a position of lesser importance than, say, precision or completeness.

The goal of the mathematical approach, for example, is to find an *exact* solution: implicitly because it is believed that this will yield the best description of behaviour. However, although we might be able to formally obtain a solution this does not guarantee that we can understand what it means. Acton & Squire share this view [Acton & Squire 85] and collate a *pot pourri* of examples from which we induced and automated one of the models of approximate reasoning in this thesis. On the particular question of comprehensibility, their text cites the following perspicacious example.

Consider, a mathematical model of a chemical reaction in which $A+2B \rightarrow C$. This is associated with the differential equation

$$dn_c/dt = (n_a - n_c)*(n_b - 2n_c)^2 \qquad (1.1)$$

which has, as an exact solution,

$$t = (1/K)[\ (1/(2n_a - n_b))((1/(n_b - 2n_c) - 1/n_b) + (2/(2n_a - n_b)^2)(\log_e[(1 - (2n_c/n_b))/(1 - n_c/n_a)]\ ] \qquad (1.2)$$

Although formally correct this formula is not very useful in practical terms. For example, it it is difficult to extract the qualitative form of the solution, the key relationships between the parameters and the effects of parameter perturbations on the timescale of the reaction. These, however, are precisely the sorts of questions which must be answered in order to grasp the meaning of the solution.

Likewise, qualitative reasoning tends to have evolved away from the issue of deriving comprehensible descriptions of behaviour: its principal goal being to predict all qualitatively distinct modes of behaviour. As these are usually couched in the form of state transition sequences or envisionment graphs they are therefore a very low level

behavioural description. The earlier work on naïve physics [Hayes 79, Forbus 84, Hayes 85, Hobbs & Moore 85] did focus on the comprehensibility of descriptions but as the techniques have matured, other criteria such as soundness and completeness having gained ascendency over this.

In practical applications, however, we are usually concerned with converting the information in a differential equation into an understanding of the key features and functional relationships of its solution. The issue of great precision is of secondary importance and a purely qualitative description of behaviour is often too weak to be of use.

Approximate functional reasoning restores comprehensibility without abandoning mathematics entirely. We take the view that mathematical descriptions of behaviour provide a rich and succinct representation language. However, there is a fine balance to be struck: if the formulae are too complex the effort required to grasp their meaning outweighs the gain in economy of representation.

We can see just how useful approximate functional descriptions can be by examining Acton & Squires approximate solution to the chemical kinetics problem posed above [Acton & Squire 85]. It is †

$$n_c \approx \tfrac{1}{2}\, n_b\, (1 - \exp(-t/\tau)) \tag{1.3}$$

where

$$\tau = 1/[K n_a n_b (1 - n_b/4n_a)]. \tag{1.4}$$

To anyone trained in basic engineering this version is packed with information and can be read almost as a linguistic description of behaviour: n_c rises from zero approximately exponentially to attain the final value $\frac{1}{2}n_b$; if n_a is increased the reaction will reach completion quicker; the timescale is inversely proportional to $n_a n_b$ modulated by a slowly varying term in n_b/n_a. These commonsense engineering inferences were buried under the complexity of equation (1.2).

† We create a computational model of Acton & Squire's theory in *Chapter 3* and extend it to yield an even better approximation than theirs.

The difference in goals makes it difficult for pure mathematicians to appreciate the style of engineers and *vice versa*. Mathematicians tend to sneer at the engineering approach as being *ad hoc*, imperfect and crude whilst engineers tend to regard the rigorous approach as unnecessarily pedantic, obscure and cluttered with trivia. Depending on one's training one is usually drawn to one camp or the other. We therefore offer a warning to those of a pure mathematical disposition: we use mathematics as a tool to aid understanding not an end in itself. We are most definitely not attempting to built a symbolic differential equation solver. Instead our aim is to model some of the processes we believe real engineers employ in finding approximate solutions.

1.2.2 Deficiencies of Other Schemes

There are other reasons, apart from improved comprehensibility, why approximate functional reasoning is worthwhile. These obtain from limitations in either the mathematical or qualitative approaches.

The mathematical approach is handicapped because the equations governing many real world situations cannot be solved exactly, in closed form. By relaxing the requirement for an exact solution however, we can find approximate solutions of these difficult equations which might yield useful information.

Moreover, even when the equations can be solved, the apparent precision is, in many cases, fictitious as the real physical system may differ from the idealisations implicit in its mathematical model. In such circumstances it would be inappropriate to expend a great deal of effort in solving the equations exactly as there is no guarantee that it would lead to significant predictive power or physical insight.

Approaching the problem from the other direction, qualitative methods have their limitations too. The behavioural predictions include neither a quantitative measure of time nor the explicit functional relationships between problem parameters that give rise to the observed behaviour. Yet, in many applications, timescale or dependency information is crucial.

An approximate functional solution, however, offers both these features, albeit in a crude fashion, and therefore yields more useful information than a qualitative technique used alone. However, it is only fair to point out that if a system cannot be specified mathematically, neither approximate functional reasoning nor exact analytic reasoning can be applied. In such cases qualitative reasoning is the best that can be done.

1.2.3 Approximation as an Aid to Comparative Analysis

A third distinguishing feature of approximate functional solutions is their suitability for answering comparative analysis queries. By stripping away a lot of the complexity of the exact solution to reveal its gross underlying features, approximate functional solutions are often good enough to estimate how specific changes to parameters will affect the behaviour of the system.

There has been work in qualitative reasoning supporting inferences pertaining to how qualitative perturbations of parameters effect the qualitative properties of a behaviour [Weld 87, Weld 88a, Weld 88b, Weld 88c]. By using approximate functional solutions it is possible to be more precise. For example, whereas qualitative comparative analysis can infer whether a period of oscillation will increase or decrease given a parameter perturbation, it is possible, using an approximate functional solution, to say by how much it is affected *e.g.* doubling a certain parameter increases the period by a factor of $\sqrt{2}$.

Moreover, approximate functional solutions can also be superior to exact solutions in this respect. This is because exact solutions can actually hinder comparative analysis by introducing too many antagonistic effects. As approximate functional solutions only deal with the gross properties of a behaviour the complicating small terms can be suppressed.

1.2.4 Cognitive Fidelity

Finally, approximate functional reasoning seems to be in closer accord with the inference practices of real engineers. Humans typically reason about systems in terms of qualitative

features (*increasing, decreasing, positive, negative*) [Stevens & Gentner 83, Kuipers & Kassirer 83], orders of magnitude (*negligible, comparable*) [Raiman 86, Dague *et al* 87] and behavioural trends (*oscillating, damped*) [Weld 86]. This information is then exploited to guide a more mathematical analysis.

In this thesis we will identify various problem abstractions which appear to be of use to real engineers and offer a computational formalization of them.

To summarize, approximate functional reasoning is motivated because:

- unlike qualitative reasoning it yields a *high level* behavioural description in the form of a mathematical function, restores a *quantitative* time scale to a process and supports *comparative* reasoning about how changes in parameters affect behaviour

- unlike exact mathematical reasoning it emphasizes *comprehensibility* and *adequacy* over precision

Thus approximate functional reasoning fills a niche between qualitative reasoning and exact analytic reasoning. We envisage using approximate functional reasoning in two settings: intelligent tutoring systems and design support. However, the current focus is on developing techniques to find approximate functional solutions rather than developing ones to exploit them.

1.3 Proposed Approach

The initial distinction we draw is really one of a change in perspective: traditionally a differential equation is seen as a mathematical specification of some system which determines a solution describing its behaviour. Instead, we consider a differential equation to be a kind of intensional knowledge representation of a behaviour and do not limit

ourselves solely to standard mathematical analyses as the only way to arrive at its solution (the extensional representation).

1.3.1 Paths to Approximation

The traditional view of the relationships between physical reality and mathematical or qualitative models was summarised in the following diagram by Kuipers [Kuipers 86a p291].

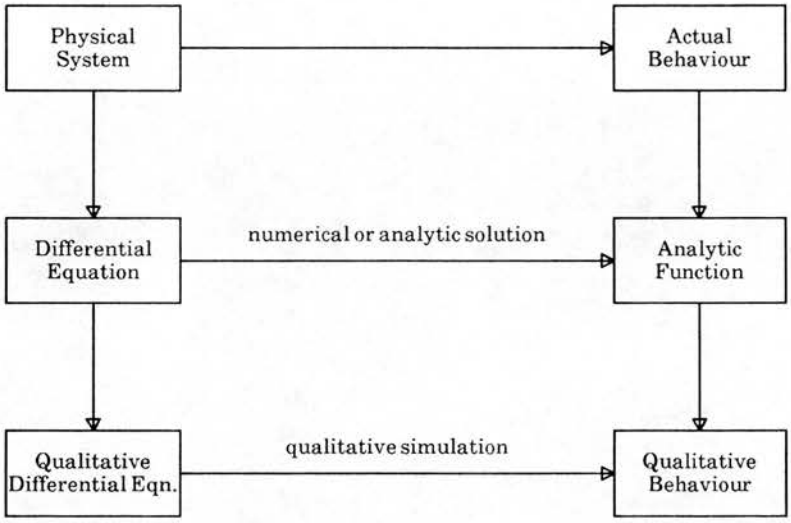


Figure 1-2. Relationship between Qualitative and Analytic Reasoning

Notice the direction of the arrows. Implicitly this diagram is saying that a qualitative behaviour is an abstraction of an exact mathematical solution but that nothing can be learned about a mathematical solution from a qualitative one. This is one of the tenets we challenge in this thesis. We believe it is possible to gain useful mathematical information but only by weakening the insistence that we map back to the exact solution. If we enlarge our scope to find an approximate functional solution in the neighbourhood of the exact solution (in a manner made precise in later chapters) we are led to a new philosophy embodied in the following diagram:

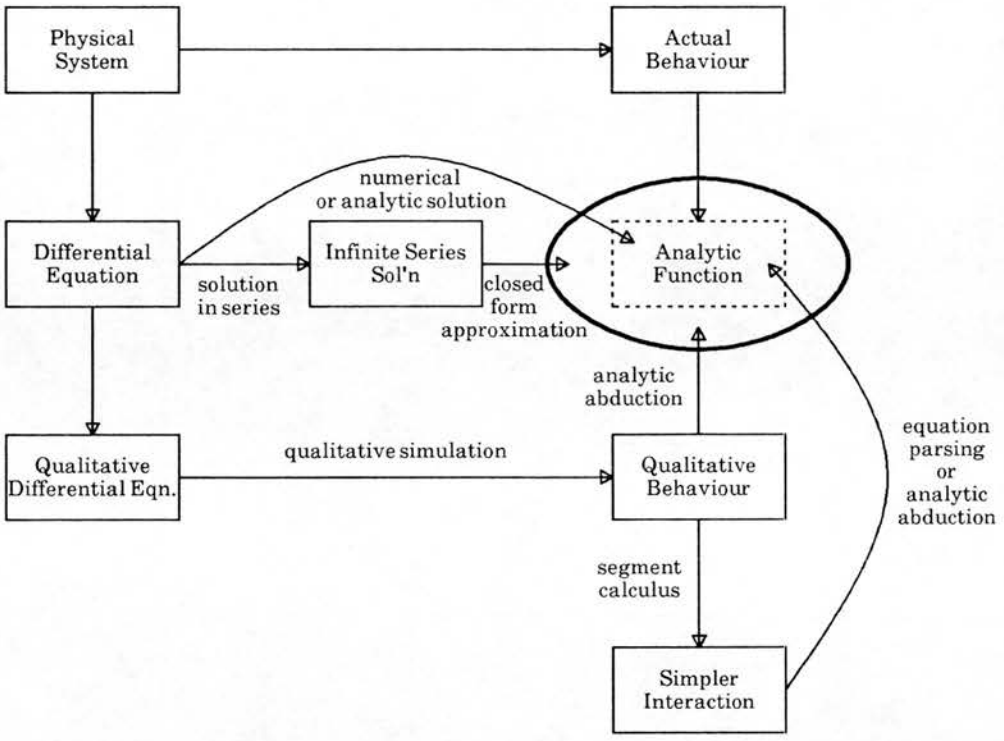


Figure 1-3. Relationship of Approximate Functional Reasoning to Other Schemes

Notice how each path now terminates either at the exact solution or in a neighbourhood about it. The new annotations on the arcs correspond to techniques to be described in subsequent chapters.

There are two basic ways to proceed. Either by solving the equation exactly and approximating the resulting expression or abstracting the equation, reasoning in the abstraction space and mapping what is learnt back to the mathematical level.

We investigate both approaches. However, because others [Sacks 85a, Sacks 85b] have already shown how to obtain closed form solutions of ordinary differential equations by automating standard methods (*e.g.* Laplace Transforms) we will avoid replication of that aspect in this thesis. This is not to say such equations are unimportant, merely that there are thin pickings down that avenue of research. What we can do however, is to pick up from where Sacks left off. In the conclusions of his papers, Sacks appealed for an exploration of equations which cannot be solved in closed form equations. This we have done in *Chapter 6*.

We emphasise that we have not overlooked the more traditional methods but we have deliberately ignored them as they contribute little new.

1.3.2 The Research Issues

The main problems to be addressed in implementing an approximate functional reasoning system are:

- *Abstraction*. What elements of a real solution can be abstracted away to make an approximate functional solution?

- *Strategy*. Is it better to create an approximate functional solution *top down* by reasoning directly from the differential equation or *bottom up* by reasoning from the equations' exact solution?

- *Adequacy*. Is a proposed approximation sufficiently accurate (in quantitative terms) for predictive purposes?

- *Fidelity*. Is it possible to ensure that the approximate functional solution is qualitatively equivalent to the real solution?

- *Optimality*. Is there a notion of a *best* approximate functional solution?

1.4 A Tour Through the Techniques

This section describes the overall structure of this thesis and the contents of subsequent chapters. The idea is that the techniques should dovetail together to achieve a style of reasoning closer to that of a physicist or engineer rather than a mathematician. The key difference being to sacrifice the goal of finding the *correct* solution to a differential equation

in favour of finding an *approximation which is adequate for the purpose to which such knowledge will be put.*

- *Chapter 2*. In order to place this work in proper perspective it is necessary to consider the different of approaches to approximating the solutions of differential equations. The three paradigms of *approximate, qualitative* and *geometric solutions* are identified. We argue that whilst other systems have exploited *qualitative* and *geometric* solutions the *approximate functional level* has been overlooked. Yet for real engineers solving real problems *approximate functional reasoning* is a respected tool to aid understanding.

- *Chapter 3*. One approach to the automation of *approximate functional reasoning* requires that the qualitative form of the solution be obtained and used to constrain the search for a parameterized analytic function of roughly the right shape. By adjusting the parameter values it is possible to optimize the conjecture with respect to the original differential equation and then to check the accuracy of the approximation by determining its sensitivity to the details of the method used to derive it. This model is embodied in the **analytic abduction** procedure. A number of examples are given and enhancements are suggested to deal with cases where the first trial function proves to be inadequate.

- *Chapter 4*. One particular way in way **analytic abduction** can fail is if there is no analytic function, known to the system, which is congruent to the qualitative behavioural prediction. **Segment calculus** addresses this problem by finding an interpretation for the observed qualitative behaviour as an interaction of two simpler qualitative behaviours.

- *Chapter 5*. At the very least **segment calculus** identifies a plausible operator decomposition of the real solution. **Equation parsing** shows how such partial information can be used to find a set of termwise interactions which when consistently and completely decoded can reveal the exact closed form solution, if it exists.

- *Chapter 6*. Many equations, cannot, as a matter of principle, be solved in closed form. For such equations it was necessary to invent a more direct method of functional approximation we called **closed form approximation**. This is based on solving the equation exactly, but as an infinite series, and then approximating this series in closed form. Efficient computation required a novel and indirect representation of infinite series based on a characterisation of its various properties as sequences of numbers represented via generators in a modular arithmetic. Various algebraic operations can then be encoded via operations on the generators.

- *Chapter 7*. A second direct method of functional approximation emerged as a solution to a sub-problem in the **analytic abduction** procedure. Often expressions were created which, though formally correct, obscured the key relationships with other minor terms. To maintain the comprehensibility requirement necessitated inventing an aggressive simplification strategy we coined **back-of-the-envelope reasoning**. This is able to approximate expressions containing parameters of unknown absolute value by exploiting ordinal relations between them to rewrite the expression into a form where small terms may be identified and thence eliminated or bounded.

- *Chapter 8*. Finally the various filaments are summarized, their research contributions re-iterated and conclusions drawn as to how each technique addresses the research issues laid down at the outset. We then conclude with suggestions for further work.

We re-iterate that the central concern of this thesis is not about solving differential equations symbolically. Instead we concentrate on the integration of qualitative and analytic reasoning to effect an approximate functional solution.

Chapter 2

Related Work

2.1 Introduction

This chapter reviews previous work relating to the approximate functional level. Although approximate functional reasoning has not been exploited by existing symbolic manipulation programs (**QMR** [Sacks 85a, 85b], **MACSYMA** [Martin & Fateman 71], **MAPLE** [Char *et al.* 85], **PRESS** [Bundy & Welham 81]) it is a respected and powerful tool used by many engineers. Hence we begin by examining mathematical approaches to the problem of approximating the solution of a differential equation as these will form the basis for the algorithms developed in subsequent chapters. We then contrast these with Artificial Intelligence techniques aimed at deriving the qualitative behaviour of solutions of differential equations. These either construct a description of the possible sequences of state transitions the system may exhibit else build a geometric representation of the system's behaviour. We also describe a reconstruction of the **QSIM** algorithm [Kuipers 85, Kuipers 86a] as this will be used in the next chapter. We conclude the chapter by outlining the advantages of approximate functional reasoning over the other inference schemes.

To begin we should be clear about the terminology we will be using.

approximate solution	an analytic solution which is close to, but simpler than, the exact solution
qualitative solution	a description of the behaviour implied by the differential equation in terms of the transitions between intervals and intervening points
geometric solution	a phase portrait
analytic solution	a continuously differentiable function
numerical solution	a table of pairs of independent/dependent variable values
piecewise solution	a set of analytic functions defined on abutting intervals with possible discontinuities at the boundaries

Unless otherwise stated, "differential equation" refers to an ordinary differential equation.

Figure 2-1 provides an overview of the relationships between a differential equation and the different types of solutions (*i.e.* extensional representations of behaviour) which may be obtained from it. The structure of this chapter mirrors this taxonomy.

The arcs, some of which are annotated with the names of Artificial Intelligence programs, are traversed in a top to bottom direction. Each possible path through the diagram corresponds to a different solution technique.

The first row of elliptical nodes partitions the techniques into those which work directly with the differential equation versus those which transform it into an "equivalent" form and then work with that. For those methods which do modify the equation, the middle row describes the kind of equation into which the original is mapped. Finally, the third row classifies the techniques required to map the (possibly modified) equation into various types of solution.

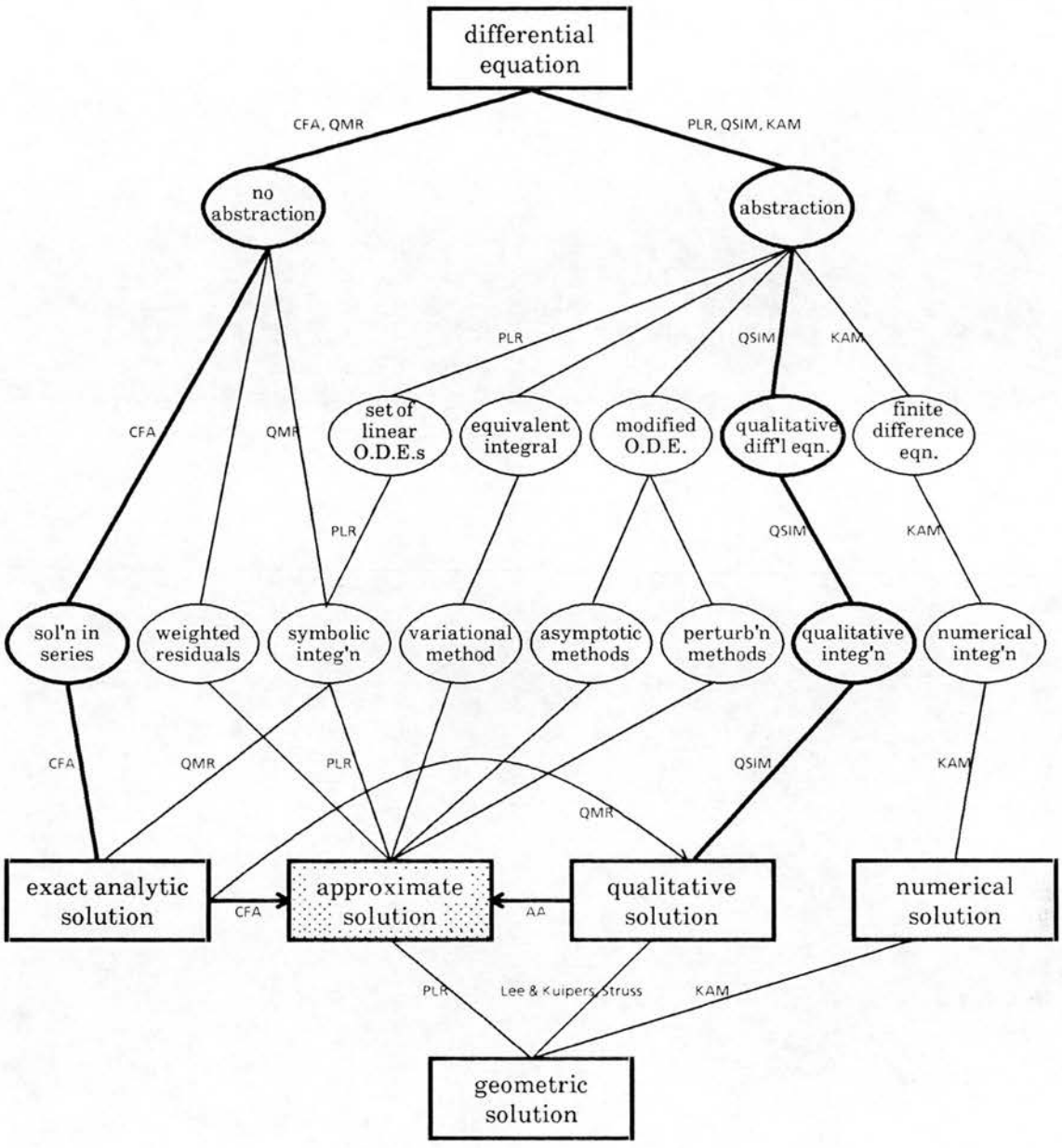


Figure 2-1. Paths to Alternative Types of Solution Annotated with A.I. Systems Implementing them

Each of the programs corresponding to the labels will be critiqued in a subsequent section. In particular, "**AA**" and "**CFA**" are our own techniques to be described in *Chapter 3* and *Chapter 7* respectively. "**AA**" is actually a synthesis of a "weighted residual" method and qualitative reasoning.

The following sections describe methods for obtaining *approximate*, *qualitative* and *geometric* solutions. For completeness, we outline methods which have not been automated as well as those which have. As we regard *exact analytic* solutions and *numerical* solutions as stepping stones to some higher level behavioural description, we will not dwell on their details. From the perspective of Artificial Intelligence, the *approximate*, *qualitative* and *geometric* types of solution are the ones of principal interest.

2.2 Approximate Solutions of Differential Equations

Few differential equations may be solved exactly in closed form. Consequently there has been considerable effort in devising approximate methods. By these we mean techniques which yield a function which is close, according to some metric, to the exact solution of the equation. This immediately excludes numerical methods which merely produce data points rather than functional relationships between parameters.

Within the class of approximate methods there are

- perturbation methods
- asymptotic methods
- variational methods
- weighted residual methods
- piecewise linear methods
- series methods

The weighted residual and series methods are, for the purposes of this dissertation, the two most important approaches to approximate functional reasoning. This is because one of the weighted residual methods provides the basis for the **analytic abduction** technique described in *Chapter 3*. Likewise, series methods, discussed in detail in *Chapter 6*, serve as a prelude to the **closed form approximation** technique developed in *Chapter 7*. However, it is instructive to consider the scope of all approximate methods, and their relationships to previous work in Artificial Intelligence, to better place this thesis in perspective.

2.2.1 Perturbation Methods

Perturbation methods [Mathews & Walker 70] address problems which are not too far from some known standard problem. The usual approach is to introduce a small variation, ε, to some variable in the exactly soluble equation and to find solutions of the equation for ε small given that we know the solution for $\varepsilon = 0$. The method is typically only useful for predicting the behaviour of slightly nonlinear systems or systems with almost constant coefficients.

For example, the equation of motion of the small amplitude linear pendulum is

$$d^2\theta/dt^2 + (g/\ell)\,\theta = 0$$

We can approximate the equation for slightly larger angles by introducing a cubic nonlinearity, $\theta^3/6$ (the second term in the Maclaurin expansion of $\sin\theta$).

$$d^2\theta/dt^2 + (g/\ell)(\theta - \theta^3/6) = 0$$

When $|\theta| < 1$ this is certainly a small variation. This equation can be solved to discover the new feature of the nonlinear oscillator: that its period depends on the initial amplitude.

2.2.1.1 Qualitative Perturbation Theory

Differential Qualitative Analysis (DQ) [Weld 87, Weld 88b] can be seen as a kind of qualitative perturbation theory. It exploits the *relative change* (or perturbation) between two systems to predict how the behaviour of one differs from that of the other. Given a structural description and a qualitative behavioural prediction we can ask what will happen to the behaviour if some parameter in the structural description is perturbed in some way. Notice that this is subtly different from simulating two related systems because qualitative reasoning cannot yield numeric temporal information. Consequently, if we simulated two similar systems independently we would have no way of knowing what the temporal relationship was between their respective time points. The key insight is to use *perspectives* to express the qualitative of state of one parameter relative to another.

For example, consider a simple spring-mass system. To answer the question "What happens to the period if the mass is increased" requires an inference to the effect that the force on the mass is "the same". Weld considers what is actually meant by this. It cannot be the same force as a function of time as there are times when the the new force exceeds the old one. In fact it means that the force is the same at the same position. Hence, force is the same from the perspective of position. The formalisation of this concept leads to a calculus of relative changes.

Ability to reparameterize a system from different perspectives is the key insight to the representation of relative change.

Advantages

- It generates explanations of why a perturbation produces a certain effect.
- It does not require an explicit function for the perturbed parameter in order to determine the effect of the perturbation.

Disadvantages

- If a change induces antagonistic effects in an expression ambiguities in the qualitative calculus may not be resolvable *e.g.* if $y = x + 1/x$ what happens if $x \uparrow$? This is unanswerable qualitatively because $x \uparrow$ and $1/x \downarrow$.
- Without extra constraints such as conservation of energy (or a problem reformulation which encodes them implicitly) it is possible to generate ambiguities *e.g.* comparative analysis cannot say anything about the *whole* period of an oscillation without conservation of energy to relate the amplitudes. Otherwise comparative analysis can only reason about the relative change in behaviour up to the first turning point of a cycle.
- There may not always be a useful perspective

2.2.2 Asymptotic Methods

Asymptotic methods are designed to find approximations in the limit as some parameter becomes very large or very small. Asymptotic solutions may be used to

- discriminate between competing guess solutions in cases were boundary conditions prescribe behaviour at a magnitude extreme
- build up the structure of the full equation by noticing that many exact solutions contain the asymptotic solution explicitly as a product with some moderating function.

For example, the Schrödinger equation

$$[-1/2\, d^2/dx^2 + 1/2\, x^2] \circ \psi = E\psi$$

with boundary condition $\psi \to 0$ as $|x| \to \infty$ can be rewritten in the simpler approximate form

$$-d^2\psi/dx^2 + x^2\psi = 0$$

when $|x| \gg E$. This has a solution up to leading powers of x as

$$\psi = \exp(\pm \tfrac{1}{2}x^2)$$

as $d^2\psi/dx^2 = \exp(\tfrac{1}{2}x^2)\,(x^2 + 1)$ or $\exp(-\tfrac{1}{2}x^2)\,(x^2 - 1)$ respectively and $x^2 \gg 1$. However, the boundary condition precludes the positive exponential.

The complete solution for this problem is found from formal methods [Landshoff & Metherell 79] to be

$$\psi_n = \exp(-\tfrac{1}{2}x^2)\, H_n(x)$$

where the $H_n(x)$ are Hermite polynomials of order n. Notice that the asymptotic form appears explicitly in the full solution as a product with some moderating polynomial.

2.2.2.1 Qualitative Asymptotic Analysis

There has been considerable work on more sophisticated asymptotic analysis in mathematical physics [Jeffreys & Jeffreys 56, Jeffreys 62, de Bruijn 58]. However, the automation still focuses on accuracy rather than comprehensibility.

In Artificial Intelligence, the work which is closest *in spirit* to asymptotic analysis is Weld's *exaggeration* technique [Weld 88a, Weld 88b, Weld 88d], although the formal machinery, being based on non-standard analysis [Davis & Hersch 72, Robinson 66], is quite different.

Exaggeration attempts to answer queries which ask for the behaviour of some system relative to another (comparative analysis) by exaggerating a perturbation, simulating the modified system, comparing the behaviour with the original system and inferring a trend from the original to the modified behaviours. An implicit assumption is that the system responds monotonically to a perturbation. If this assumption is false exaggeration can make erroneous predictions.

Weld has been careful to ground his theory in non-standard analysis. Like **QSIM** [Kuipers 85, Kuipers 86a], the qualitative state of a parameter, P with landmarks $p_0 < ... < p_k$, consists of a pair $\langle HR\text{-}qval, HR\text{-}qdir \rangle$ [†]: the difference being that the qualitative values used are abstractions of hyper-real numbers rather than real numbers. In particular, $HR\text{-}qval = X$ where

$$X ::= inf \mid p_i \mid (\text{HALO } p_i +) \mid (\text{HALO } p_i -) \mid \{p_i, p_{i+1}\} \mid \{p_k, inf\}$$

("|" signifies "or") and $HR\text{-}qdir = (Y,Z)$ where

$Y ::= inc \mid std \mid dec$ (sign of a parameter's derivative)

$Z ::= inf \mid fin \mid negl \mid 0$ (hyper-real order of magnitude of the derivative)

In addition, Weld introduces a more refined (hyper-real) temporal representation which has 0, *negl*, *fin* and *inf* time intervals (for *zero*, *negligible*, *finite* and *infinite* respectively) instead of just instants and intervals. The overall simulation algorithm, **HR-QSIM**, is similar to that of **QSIM** but the finer qualitative state description and richer temporal representation

[†] see page 35 for exaplanation of $\langle qval, qdir \rangle$ pairs

allows **HR-QSIM** to reason about the time a parameter may exist in a state before making a transition. This permits two new filters to be added to the standard **QSIM** set (see §2.3.2) called *predecessor-persistence* filters and *successor-arrival* filters. These can be used to prune inconsistent successors or choose between competing possible transitions.

The properties of a calculus of non-standard qualitative values can be quite perplexing. For example, as Weld says "it may take longer for a parameter to transition to its new value than it spends in its old value". Clearly, the transition through haloes surrounding landmarks is non-trivial. Moreover, the use of non-standard analysis is a long way from the manner humans reason about expressions at magnitude extremes. It is forced upon Weld because he wanted to extend qualitative reasoning to deal with both infinities and infinitesimals in a mathematically sound way.

In *Chapter 8* we introduce an alternative way of performing a kind of asymptotic analysis we term "Back Of The Envelope Reasoning". This exploits the fact that the approximate functional solutions, generated by the techniques in this thesis, are explicit mathematical expressions which may therefore be subjected to standard mathematical manipulations such as limit evaluation. This leads us to a conceptually simpler system, **BOTHER** (for Back Of The Envelope Reasoner), which we use in conjunction with the **analytic abduction** technique to be discussed in the next chapter.

BOTHER is a rewrite rule system combined with a limit evaluator which can compute approximations to mathematical expressions at any magnitude extreme. It differs from other rewrite rule systems by working a problem hard *i.e.* reducing an expression containing symbolic constants of unknown absolute value to a number if possible. This either occurs through cancellation/combination rules or else by reasoning about an expression's variance and picking a mid-point value.

A final criticism of *exaggeration* is that it may not always be appropriate to exaggerate a parameter to an infinite value. For example, suppose we wanted to perform an exaggeration analysis of a simple, not necessarily linear, pendulum. We could certainly exaggerate its length, determine the period and infer a trend. But what if we wanted to examine the effect of differing initial displacements? In this case we would not exaggerate to infinity but to

$\pi/2$. **BOTHER** can evaluate expressions in *any* limit and is in this sense more flexible. However, the same presumption of monotonic system response applies to **BOTHER** too.

2.2.3 Piecewise Linear Methods

Nonlinear differential equations are notoriously difficult to solve as there are no generally applicable solution techniques. As general solution methods *do* exist for *linear* equations (*e.g.* **QMR** [Sacks 85] uses Laplace Transform → Partial fractions expansion → inverse Laplace Transform) we may construct an approximate solution to the nonlinear equation by piecewise linearising its coefficients, solving each of the linear equations on its subdomain and concatenating the results.

The key idea is to approximate the nonlinear coefficients in the equation by piecewise linear functions. For example the function $y=x^2+x$ can be approximated by two straight lines (figure 2-2(a)).

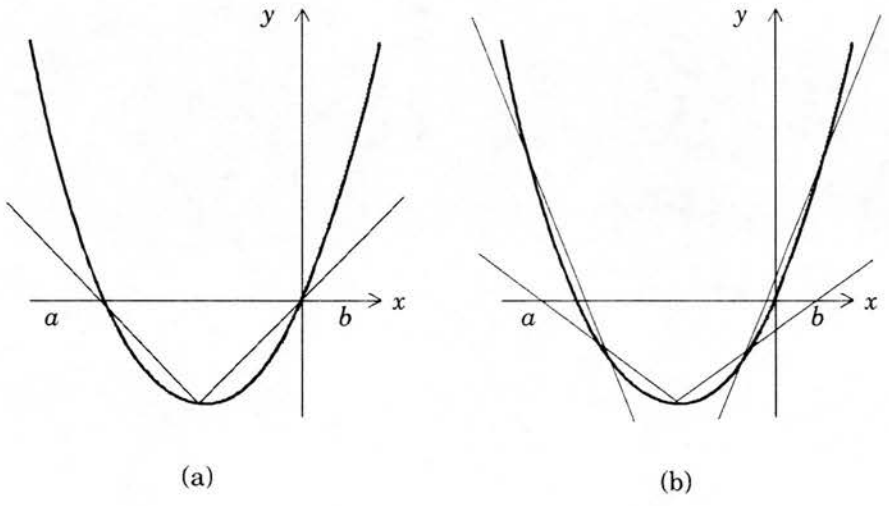


Figure 2-2. Piecewise linear approximations

If greater accuracy is required further subdivisions of the domain can be introduced (figure 2-2(b)) and piecewise linear approximations made for these too.

Sacks has automated this kind of piecewise linear reasoning in his **PLR** system [Sacks 87b, 88]. We will defer further description for the moment as Sacks' ultimate goal was to construct a representation of the phase portrait of a system of differential equations. Consequently, this work is more appropriately categorised under "geometric solutions" than simply approximate ones (see §2.4.5).

Piecewise linear solutions consist of a set of approximate solutions over abutting intervals. In general, the solutions do not join smoothly at the interval boundaries. By contrast, **analytic abduction** to be introduced in *Chapter 3*, will aim to find an approximate functional solution over the entire domain. However, **closed form approximation** to be described in *Chapter 7* is more similar to piecewise linear reasoning in that it too finds a solution over a restricted domain.

2.2.4 Variational Methods

Variational methods find an approximate solution to a differential equation by finding the conditions for a functional, related to the original differential equation, to be stationary. Hence they do not tackle the differential equation directly. A suitable functional might be a variational integral. For example from the domain of quantum mechanics, the Schrödinger wave equation (a differential equation) is equivalent to the variational integral

$$E[\phi] = <H\phi \mid \phi > / < \phi \mid \phi >$$

where ϕ is a guess at the solution of the Schrödinger equation containing adjustable parameters β_i, H is the Hamiltonian operator and $< \mid >$ is an inner product. Theory tells us $E[\phi] \geq E_0$, the lowest energy eigenvalue, with strict equality when $\phi = \psi_0$ the ground state wavefunction. We can therefore find approximate solutions for the ground state wavefunction and eigenvalue by adjusting the parameters so that this functional is a minimum. Mathematically, this amounts to solving

$$\partial E[\phi]/\partial \beta_i = 0$$

for the β_i. This yields a minimum energy eigenvalue E_{min} which when substituted into ϕ gives us an approximation to the ground state solution of the Schrödinger equation ψ_0.

Finding a functional can be difficult if not impossible (*e.g.* there is no variational principle for Navier Stokes equations including viscous and inertial terms [Finlayson 72 p286]). Moreover, problems which can be tackled using variational principles often have a classical solution by other means [Finlayson 72 p211]. However, the method has been greatly used in quantum mechanics to calculate approximate wavefunctions for atoms and molecules.

Kuipers mentions the use of integral equations instead of differential equations in [Kuipers 86c (unpublished)] but this remains to be developed.

2.2.5 Weighted Residual Methods

Weighted residual methods attempt to approximate the true solution of a differential equation by guessing that it can be written as a sum of orthogonal functions each containing adjustable parameters. By finding optimal values for the adjustable parameters the difference between the trial function and true solution can be minimised in some sense. This optimisation is usually performed indirectly by maximizing the fit of some trial function with respect to the differential equation and its boundary conditions.

The error produced when a trial function is substituted into a differential equation is called the equation residual, R_e. Had we guessed the true solution then this residual would be identically zero throughout the domain but in general it will be non-zero. Various criteria exist for minimizing the R_e all of which can be cast as the problem of minimising a weighted integral involving the equation residual over an interval (a,b). This involves finding values for the adjustable parameters such that

$$\int_a^b w_i R_e \, dt \,/\, \int_a^b w_i \, dt = 0$$

Different weighted residual procedures arise by choosing different weighting functions. These include

- collocation
- subdomain
- Galerkin
- least squares

The members of the approximating series are called *trial functions* and are usually taken, in order, from a family of orthogonal functions. For most of the methods described below, this device results in some simplification in the computations.

2.2.5.1 Collocation

Collocation employs Dirac delta functions [Landshoff & Metherell 79] as the weighting functions. This choice neatly sidesteps the need for explicit integration as

$$\int_a^b \delta(t - t_i)\, R_e(t)\, dt = R_e(t_i)$$

The points t_i are known as collocation points and they are chosen arbitrarily from the interval (a,b). For collocation we simply have to find values for the β_i such that at each point t_i, $R_e(t_i) = 0$. To do this we must choose as many points t_i as there are adjustable parameters in the trial function so that we will obtain enough simultaneous algebraic equations to solve for the β_i. For each choice we assume that on substitution into the differential equation the residual at this point will vanish and find values for the β_i such that this is the case. This method is based on the implicit assumption that the exact solution does not stray too far from the trial function in between the collocation points.

2.2.5.2 Subdomain

In the subdomain method we again select a simple weighting function, this time a step rather than a spike, which selects out points over a subdomain rather than at single locations. Partition the domain, D, into as many subdomains as there are adjustable

parameters. If there are n adjustable parameters we may call these subdomains D_j, $1<j<n$ and define the weighting function to be

$$w_j(t) = 1 \text{ if } t \in D_j$$

$$w_j(t) = 0 \text{ if } t \notin D_j$$

With this choice of weighting function the integral of the residual over each subdomain is set equal to zero to yield n simultaneous equations for the β_j.

2.2.5.3 Galerkin

In the Galerkin method the weighting functions are taken from the same family as the basis functions in the trial solution.

$$w_i(t) = \phi_i$$

To understand this method for residual minimization we must recall two things. First a set of functions $\{\phi_i\}$ is complete if any function can be expanded in terms of its members *i.e.* $f = \Sigma\ a_i\phi_i$. By choosing $w_i(t) = \phi_i$, the weighting functions become elements of the same complete set as that used to expand the solution. Second, a continuous function is zero if it is orthogonal to every member of a complete set. Hence the residual is forced to be zero (minimized) by making it orthogonal to each member of a complete set of functions.

The Galerkin method is usually assumed to be superior to collocation at least. However, in practice it is possible that it might require a significant number of basis functions to achieve the desired accuracy. Hence, the Galerkin method in not *necessarily* the most concise of the weighted residual methods in any given application. However, empirical evidence suggests that it often is.

2.2.5.4 Least Squares

The Least Squares procedure is a variational method in disguise. If we defined the functional

$$I(\beta_i) = \int_a^b R_e^2 \, dt$$

and demanded that this be stationary then this is equivalent to

$$\partial I/\partial \beta_i = 2 \int_a^b R_e \, \partial R_e/\partial \beta_i \, dt = 0$$

By taking weighting functions

$$w_i(t) = \partial R_e/\partial \beta_i$$

this matches the general form for weighted residual minimization. Thus in this procedure the integral of the square of the residual is minimised with respect to the undetermined parameters to provide n simultaneous equations for the β_i.

2.2.5.5 Comparison of Weighted Residual Methods

Collocation is the simplest method to automate because it avoids the need for explicit integration. All the other methods involve integrals which become progressively more difficult in the order presented above.

The choice of collocation points is arbitrary but it is customary to choose an even distribution over the domain. If we use orthogonal polynomials as the basis functions it is convenient to choose collocation points as the roots of successive members of the family.

Ames notes [Ames 77 p256] that collocation is often overlooked in favour of its more sophisticated cousins. Yet, in the absence of any general error analysis, the presumption of its *necessary* inferiority is unfounded.

There are three approaches to choosing trial functions:

- *Interior Method.* Choose trial functions which satisfy the boundary conditions identically. Only the interior equation residual is then non-zero.

- *Boundary Method.* Choose trial functions such that the differential equation is satisfied. The only errors are the residuals at the boundary and initial conditions.

- *Mixed Method.* Choose trial functions which satisfy neither the boundary conditions or the equation identically. The error is distributed over both the equation and boundary residuals.

As all the weighted residual methods rely initially on guessing a suitable trial function, the method seems ripe for some assistance from artificial intelligence. An arbitrarily accurate approximation is obtained by including sufficient terms in the approximating series. However if we made a particularly bad choice of trial function the approximating series may require 10 or more terms to achieve the desired accuracy. Thus although we may have solved the problem in principle the question arises as to whether it is *practicable.*

A better approach might be to reappraise one's choice of trial function. Ideally, we should like to find a family for which a single trial function was sufficient to meet the required accuracy. This raises new questions, so far overlooked in the mathematics literature, about how to automate navigation through the space of possibilities.

When choosing suitable trial functions one must consider:

- *simplicity*: the trial function should be sufficiently simple to facilitate symbolic manipulation

- *coverage*: the adjustable parameters should be inserted such as to give a broad range of behaviour simply by varying their values

- *boundaries*: it may be desirable to choose functions which can accommodate the boundary and/or initial conditions

2.2.6 Series Methods

Not all equations have solutions which can be written in closed form. Series methods expand the solution as an infinite series about some point. Usually there is high accuracy in the vicinity of the initial state. Series methods are explained fully in *Chapter 6* as they form the basis for the **closed form approximation** technique described in *Chapter 7*.

2.3 Qualitative Solutions of Differential Equations

The previous section concentrated on finding approximate analytic solutions to differential equations. We contrast this work with recent advances in artificial intelligence aimed at finding qualitative solutions *i.e.* descriptions of the behaviour of solutions of which preserve all important distinctions but which do not involve explicit functions. Consequently, *qualitative* solutions are quite distinct from, and much weaker than, *approximate* solutions. Their usefulness arises because they can be derived from correspondingly weaker system specifications.

In the following sections we classify the research according to the underlying *strategy* rather than by author or chronological development. This has the advantage of placing the various systems in sharper focus and exposing the niche we are carving out for approximate functional reasoning. Each section begins with an outline of the procedures comprising a certain strategy. This is followed by a description of the particular strategy's advantages and disadvantages and a critical review of a system which implements the strategy.

2.3.1 Symbolic Integration + Qualitative Abstraction

The first strategy we consider is to solve the differential equation exactly to obtain an analytic solution and use this to construct a qualitative solution.

Advantages

- There are no spurious behaviours.
- Both analytic and qualitative solutions available.
- Coefficients may contain non-numeric constants.
- No qualitatively distinct behaviour is missed.

Disadvantages

- The functional form of the coefficients must be known *i.e.* it is not sufficient to know that some coefficient is a monotonic function of some parameter.
- Few differential equations can be solved exactly. Consequently, the applicability of this strategy is severely limited.

Systems: QMR

Sacks' **QMR** system falls into this category [Sacks 85a, Sacks 85b]. **QMR** uses the method of Laplace transforms [Kreider *et al.* 80] to obtain the exact solution of a subset of linear second order ordinary differential equations and a program, **QM**, for constructing qualitative descriptions of them.

QM has knowledge of certain primitive functions including cubics, quadratics, polynomial-exponential products, exponentials, logarithms, trigonometric and hyperbolic functions, their inverses, exponential-trigonometric products, triangular functions and step functions [Sacks 85b p47].

Each of these (at least piecewise continuous) functions is represented by a sequence of *fun-ints* defined over interval between $[lb,ub]$. A function is strictly monotonic or constant on the interior of each *fun-int* and the end points correspond to discontinuities or domain boundaries. A fun-int is essentially a frame having slots for lb, ub, $f(t)$, $f(lb)$, $f(ub)$, $f'(t)$, $f''(t)$,

directionality, convexity, $f^{-1}(t)$ all represented in explicit analytic terms. For example, Sacks represents $\sqrt{x}$ on $[0,\infty)$ as follows

Property	Value
direction	up
lb	0
ub	∞
fun	$\sqrt{x}$
lb-val	0
lb-r-lim	0
ub-val	NIL
ub-l-lim	∞
singularities	NIL
inverse	x^2
derivative	$1/(2\sqrt{x})$
der2	$-1/(4x\sqrt{x})$
convexity	$((0 - 1\ \infty))$

Table 2-1. Example of QMR's Representation of Functions

The last entry in Table 2-1 corresponds to ((lb sign-of-der2 ub)). The others are self-explanatory. Periodic motion could be represented by a finite number of *parameterised fun-ints* (see *Chapter 4*). The information contained in each *fun-int* is sufficient to sketch the function although some of the slots may not be filled *e.g.* not all functions have an inverse.

The qualitative description is accomplished by parsing the analytic solution top down until functions are reached which match templates of the primitives in the library. *Instantiation* algorithms then select appropriate constants such that the matching library primitive can be made equal to the corresponding subterm from the solution. These values are then propagated through the slots in the **FD** to amend the entries *e.g.* derivatives, bounds and inverse, appropriately. Next composition algorithms combine the individual **FD**'s of the subterms into a composite **FD** for the whole expression. Finally the composite expression is summarised to extract higher level information *e.g.* directionality, convexity, discontinuities extrema, limits, singularities and asymptotes. Some of these arise directly as entries in the **FD**, others require further (but straightforward) mathematical analysis such as numerical evaluation. In the original version [Sacks 85a, Sacks 85b], the highest

level description was a linguistic summary of a behaviour. However, Sacks subsequently developed a sketching (*N.B.* sketching not plotting) program capable of driving a graphics display [Sacks 87a].

Review

In spirit, QMR is perhaps closest to the approach advocated in this thesis. However, the approximate functional level allows us to "solve" (*via* **analytic abduction** (*Chapter 3*)) equations not amenable to exact methods and to "summarise" (*via* **closed form approximation** (*Chapter 7*)) open form solutions. Sacks was certainly aware of the need to extend the analytical capabilities of his system and mentions the two issues we address, namely *unwieldy* and *open form* solutions [Sacks 85a p139], but subsequently pursued a phase plane approach. This suppresses these issues by emphasising the trajectories of the solutions in the phase plane rather than the solutions themselves. Our approximate functional approach tackles them head on.

We certainly agree with Sacks on a number of key methodological points:

- If mathematical methods are applicable we can usually gain more inferential power by using them as opposed to a purely qualitative approach.
- The objections of the qualitative reasoning community, that it is not always possible to write down the functional form for coefficients, is valid but not damning. If a model is not completely specified the conventional approach is to pick a plausible functional approximation for an unknown coefficient by using, for example, dimensional analysis.
- An analytic approach is superior to a numerical approach which can miss important behavioural modes, be expensive and lead to false generalisations.

However we disagree in one crucial respect:

- Exact solutions are not always the most appropriate. For example, recall the solution of the bimolecular reaction equation in §1.2.1.

2.3.2 Qualitative Abstraction + Qualitative Integration

An alternative strategy is to abstract the functions and operators of the original equation to weaker forms and solve the modified problem. In **QSIM** this abstraction process maps the ordinary differential equation to a qualitative differential equation. The modified equation is then integrated using a qualitative calculus whose rules, whilst derived from conventional calculus, import new and counter-intuitive possibilities.

Advantages

- Can handle qualitative differential equations directly *i.e.* systems whose differential equation contains coefficients whose functional forms are only known to be monotonic and nothing more specific.
- Output envisionment includes all realizable behaviours of the real system
- Clear semantics.

Disadvantages

- Lose absolute time scale: only have relative ordering of time points.
- Lose absolute parameter values. If parameters are the same sign it is still possible to order them. However, if they are of opposite signs it is not possible to determine their relative magnitudes.
- Cannot currently accommodate differential equations whose coefficients are functions in the independent variable (*i.e.* for **QSIM**, differential equations involving time dependent terms).

- As the qualitative abstraction is the abstraction of many dissimilar equations behaviours can be generated which do not correspond to those of the original differential equation.
- Worse still, spurious behaviours can be generated that do not correspond to the behaviour of any of the systems sharing a common qualitative abstraction [Kuipers 86a].
- The qualitative calculus admits arithmetically impossible solutions to qualitative constraint equations [Struss 87, 88a, 88c] and introduces ambiguities absent in the equivalent equations over the reals.
- Branching factors can swamp simulations with too many possibilities [Kuipers & Chiu 87]
- Behavioural description is in terms of state transition sequences which is a very low level representation.
- Provides no insight as to the functional relationships between system parameters.

System: QSIM

QSIM is a qualitative reasoning system implementing the qualitative abstraction/qualitative integration strategy. The technique of **analytic abduction** which we introduce in the next chapter uses our own reconstruction of **QSIM** in Prolog so it is useful to describe its operation in some detail here.

Any qualitative reasoning system can be characterised by its *structural description, quantity space, notion of qualitative state, representation of time, state transition rules, behavioural description* and *inference procedure.*

In **QSIM** the *structural description* consists of a set of parameters and a set of constraints between them. A parameter is a continuously differentiable function with a finite number of critical points in any finite interval.

The constraints are of three types: arithmetic (**add**/3, **mult**/3, **minus**/2), derivative (**deriv**/2) and monotonic (**M+**/2, **M-**/2). The latter attest to there being a monotonic relationship between two parameters whilst suppressing precise functional details. In our version of **QSIM** these are derived automatically by parsing an ordinary differential equation and building the constraint set bottom up from the parse tree. In the original, these were hand coded. Alternatively, they can be typed in directly, if the system specification is only weakly known, although, in this case, analytic abduction cannot be applied to conjecture an approximate solution as this requires an ordinary differential equation rather than a qualitative one. The arguments of all the constraints take on qualitative, rather than numeric, values. Monotonic constraints allow many different functions to be mapped into a simple set of constraint equations. However, this necessarily means that some information is lost. In fact any qualitative differential equation is the abstraction of many quantitative differential equations.

Associated with each parameter are a set of *landmark* values where something interesting happens in qualitative terms *e.g.* an extremum is reached or a phase transition occurs and new structural description applies. Each landmark set always includes zero, the values of the parameter at the end points and the value of the parameter at each of its critical points (where its first derivative is zero).

The *qualitative state* of each parameter *f* is specified by an ordered pair <*qval*,*qdir*>: where *qval* is the qualitative value of the parameter (either a landmark value or interval between landmarks) and *qdir* is the qualitative direction of change (the sign of the first derivative with respect to time of the parameter, with values **inc**, **std**, or **dec**)

The *qualitative system state* at any time point or over any time interval is just the tuple of qualitative states of all system parameters at the corresponding time.

Sometimes information is known about a particular instance of a constraint which can be used to tie the landmark sets of different parameters together. These are known as *corresponding values* and can be used to check that the magnitude of the arguments of a constraint are qualitatively consistent.

Time is represented as a succession of points and open intervals. The initial state is designated time point `t0` and whenever anything interesting happens to *any* parameter a new time point is created. Thus the distinguished time points for the whole system are the union of the distinguished time points of each parameter. This means

- **QSIM** creates a global state description at every time point
- some parameters will have the same qualitative state for a sequence of time points.

Given an initial qualitative state for a parameter and an initial time point **QSIM** applies rules for determining all possible successor states thus producing a tree of qualitative states.

In essence, the transition rules state that if some parameter is increasing on some interval then it cannot be decreasing on the next interval unless it goes through a turning point at the intervening time point. Transition rules are simply qualitative statements of mathematical theorems from differential calculus under the assumption that all parameters are continuously differentiable.

An important feature is that new landmarks may be dynamically discovered as the simulation proceeds. This means that **QSIM** is able to distinguish, for example, between constant amplitude oscillation and damped oscillation (given energy constraints to restrict the envisionment). Other qualitative reasoners cannot do this.

Each node in the tree represents the qualitative state of the system at a time point or over a time interval. Thus a possible behaviour is the sequence of nodes on any path from root to leaves.

In the original algorithm it was necessary to perform pairwise filtering explicitly. In Prolog this is unnecessary: we simply use the same variable name in the appropriate argument position and it can never be assigned inconsistently.

Algorithm 2-1. `QSIM`

Input

1) a set of function symbols $\{f_1, f_2, ..., f_n\}$
2) a set of constraints involving the f_i, $i = 1, 2, ..., n$
3) a set of landmark values for each f_i, $i = 1, 2, ..., n$
4) optional range for some f_i
5) an initial time point `t(0)`
6) an initial qualitative state for each f_i, $i = 1, 2, ..., n$

Method

1) Create an *agenda* of states whose successors are desired (initially there is one).
2) Select a state from the *agenda*
3) For each parameter, determine all possible locally consistent successor qualitative states. Local consistency means being both *direction of change consistent* and *qualitative magnitude consistent*.

   Direction of change consistency involves inspecting the signs of both the parameter and its first derivative in the post-transition state. For each type of constraint there are rules delimiting the possible combinations of sign *e.g.*

   ```
   valid-dir-of-change(mult(<-,inc>, <-,inc>, <+,dec>)).
   ```

   Qualitative magnitude consistency tests the relative magnitudes of each parameter before and after the transition and applies tests to see if the tuple of parameters mentioned in a constraint could have simultaneously undergone their respective changes of magnitude. For example, if `>1` is interpreted as meaning something of magnitude greater than one then

   `valid-qual-mgtde(mult(>1,>1,>1))` succeeds

   whereas

   `valid-qual-mgtde(mult(<1,<1,>1))` fails *etc.*
4) Check that parameters common to many constraints have been assigned the same transition (pairwise consistency)
5) Apply global filters to check for *redundancy* (*i.e.* same state as immediate predecessor), *cycles* (all landmark values, qualitative values and direction of changes have occurred before on the path - mark as a cycle and don't add to agenda), and a *termination* condition (if the value of any parameter diverges to infinity - continuous differentiability guarantees that there is no successor so do not add it to the agenda)
6) Add new state to *agenda* and return to step 2)

Our reconstruction of **QSIM** fully automates the translation from the differential equation into a minimal constraint set and also revises the output qualitative behavioural prediction by propagating knowledge of discovered landmarks back through earlier states. The reason for the latter operation will be explained in *Chapter 3*.

2.3.3 Algebraic Manipulation of Qualitative Equations

At this point we have only considered the use of qualitative constraint equations for the purpose of behavioural envisionment *via* qualitative simulation. Recently, however, there have been attempts to "solve" or "transform" qualitative equations by direct algebraic manipulation. It is important to realize that combining qualitative equations is not as straight forward as conventional symbolic manipulation as, in general, they do not obey the field axioms of the real numbers [B. Williams 88, Struss 87, 88a, 88b, 88c]. For example, de Kleer & Brown's confluences [de Kleer & Brown 84] are qualitative equations over the qualitative value set $\{0,+,-,?\}$ with "$\approx$" the relation of "sign compatibility" (*i.e.* $+ \approx +$ is true). Crucially, this cannot be equality as "?" is sign compatible with anything (*i.e.* $+ \approx ?$ is also true). What this means, in practical terms, is that we cannot use cancellation or substitution as freely as we do in arithmetic.

Williams [B. Williams 88] defines a new qualitative algebra, Q1, which combines the algebras of qualitative values, $S' = \{-,0,+,?\}$, and real numbers, $\mathbb{R}$. Q1 is defined as the structure $\{\mathbb{R} \cup S', +, *, \oplus, \otimes, [\ ]\}$ where, + and * are standard arithmetic operators, $\oplus$ and $\otimes$ are qualitative equivalents and "[ ]" denotes "the sign of". Williams shows that the hybrid algebra lacks certain properties of the reals (*e.g.* it has no additive inverse), shares many (*e.g.* qualitative multiplication and unary negation are much as for the reals) and has novel properties not possessed by the reals (*e.g.* all expressions raised to a positive/negative odd power are equivalent, likewise for even powers). Having proved these properties it is possible to define a rewrite rule system for Q1 which supports simplification, canonicalization and factorization. This system is then used to perform the manipulations required by the design application [Williams 87].

Advantages

- The hybrid algebra, Q1, facilitates reasoning about incompletely specified systems.
- Q1 is a more expressive representational language than purely qualitative differential equations.
- It is particularly suited to design and verification tasks.
- However, it could also support behavioural prediction by solving a hybrid equation for some unknown.

Disadvantages

- Q1 offers a less expressive representational language than ordinary differential equations.
- Consequently, any behavioural description must still be weaker than an approximate functional one.
- The meta-level control of the rules is not discussed.

The key contribution to qualitative reasoning is to suggest that instead of reasoning about qualitative values or numbers we should reason about the *composition of qualitative relations*.

A variation of this idea is also suggested by Dormoy and Raiman [Dormoy 88, Dormoy & Raiman 88]. Their technique consists of *assembling* sets of confluences together *i.e.* reducing the number of confluences required to specify a device by discharging a shared variable. In this way they obtain relations expressing internal variables as qualitative functions of the input. The authors call the rule of combination "the qualitative resolution rule" because of its similarity with resolution in propositional logic (expressed in binary equations). It has the following form:

$$\forall x,y,z,a,b \in \{0,+,-,?\} \quad (x \oplus y \approx a \wedge \ominus x \oplus z \approx b \wedge x \neq ?) \rightarrow (y \oplus z \approx a \oplus b)$$

where x,y,z,a,b are qualitative variables, $\{0,+,-,?\}$ are qualitative values, $\oplus$ is qualitative addition, $\ominus$ is unary negation and $\approx$ is "sign compatibility".

Using qualitative resolution the authors demonstrate how a simulation task can be reduced to a value propagation task in the "assembled" confluences. The hope is that, by assembling confluences, larger qualitative models may be tackled.

A problem with the this technique is that the number of potential applications of the qualitative resolution rule grows the more it is used, leading to combinatorial explosion. So the question arises as to which of the many possible applications to pursue? A solution is proposed based on a heuristic called the *joining rule*. This rule restricts the application of resolution to pairs of confluences for which the discharged variable is unique *i.e.* there is no third confluence mentioning this variable. The joining rule guarantees that the resulting confluence is a proper model for the composed components. Algorithms are then presented for the recursive application of the joining rule in the assembling process together with four simplification rules for confluences.

Advantages

- Confluences facilitate reasoning about incompletely specified systems.
- Confluence assembly makes it possible to simulate systems which are larger than those tackled hitherto by transforming the initial system of qualitative constraints into a smaller, equivalent set.

Disadvantages

- The behavioural representation is weaker than an approximate functional description.
- It is possible, as the sizes of the systems considered increase, that there will be a combinatorial explosion of legal applications of the resolution rule which will be untamable using the "joining rule" alone.

This is a deep result intimately connected with the causal ordering ideas of Iwasaki [Iwasaki 88, Iwasaki & Simon 86a, Iwasaki & Simon 86b]. Causal ordering is claimed to be

a way of deriving a causal structure amongst a set of variables from a purely acausal set of equations. Recent work by Struss [Struss 87, 88a, 88c], however, shows how the causal structure obtained for equilibrium systems is highly sensitive to the precise mathematical formulation of the equations. Nevertheless, it is interesting that there is a convergence of ideas between Williams, Iwasaki, Dormoy and Raiman. As yet the scale up to large systems suggested by Dormoy remains to be seen.

2.4 Geometric Solutions of Differential Equations

2.4.1 Spurious Behaviours

The original version of **QSIM** [Kuipers 85, Kuipers 86a] generated behavioural predictions which provably included all realizable behaviours. Unfortunately, it also predicted behaviours that were physically impossible. Such spurious predictions limited the power of the reasoner.

Various solutions were proposed to chip away at the problem. Changing the problem specification (by even as innocuous a measure as changing the associativity grouping of parameters) changed which spurious behaviours were predicted [Struss 88a, 88b, 88c, Kuipers 86a, 88]. As each reformulation was known to predict all realizable behaviours, it would be possible to eliminate some spurious behaviours by identifying those behaviours missing from the intersection of envisionments. Unfortunately, there was no guarantee that all spurious behaviours had been eliminated.

A second approach was to introduce additional domain specific or mathematical constraints. Domain specific constraints are usually statements of domain invariants *e.g.* conservation of energy or additional system properties. However, these were still found to be inadequate to restrict the envisionment.

Further mathematical constraints were introduced based on the sign of higher order derivatives. If we think of the simple case where a parameter is increasing, (↑), but its second derivative (or curvature) is negative we know that after the critical point the parameter will decrease, (↓). However, without such knowledge the system can branch three ways.

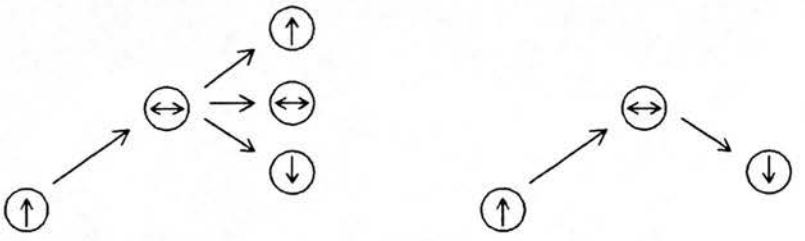


Figure 2-3. Effect of Higher Order Derivative Constraints

The curvatures of all parameters other than those of the highest order derivatives are known. So the problem arises with these highest order derivatives. Kuipers and Chiu were able to find a way of deriving an expression for the curvature of these highest order derivatives in terms of the other parameters in the system. Once these are known the transitions of the highest order derivatives are no longer ambiguous.

To obtain these signs of curvatures Kuipers and Chiu [Kuipers & Chiu 87] used a simple rewrite rule system over expressions involving qualitative values. These recursively unfold an expression for the sign of the curvature of each highest order derivative until every subterm is either an exogenous variable or linked directly to its derivative.

Even curvature constraints, however, proved to be inadequate.

2.4.2 Geometric Theory of Differential Equations

The persistence of these spurious behaviours inspired the exploitation of an alternative formalism: the **geometric theory** of differential equations [Braun 78, Lefschetz 77]

(elsewhere known as "qualitative theory" – we choose the older title to avoid confusion). The key insight is that it is possible to represent the behaviour of a dynamic system as a trajectory in a so-called phase space. Only certain shapes of trajectory are legitimate (*e.g.* trajectories may not cross) and all trajectories of the same shape correspond to qualitatively similar behaviours. By reasoning about the topological properties of predicted trajectories it is possible to identify spurious behaviours by spotting when a proposed trajectory violates the rules for the arrangement of trajectories in the phase plane. The essential difference is that trajectories in phase space encode global properties whereas the state transition sequences are governed by purely local rules and it is possible for parts of independently legitimate behaviours to become meshed together to form a spurious behaviour.

2.4.2.1 Phase Spaces

A system of first order differential equations whose coefficients do not involve time can be written as

$$dx_i/dt = g_i(x_1, x_2, ..., x_n), \quad i = 1, 2, ..., n$$

The simple case where the are two such equations ($n=2$) is informative. If we formed the ratio

$$(dx_1/dt)/(dx_2/dt) = g_1/g_2$$

this defines dx_1/dx_2 as a function in x_1 and x_2. The integral curves of this equation describe some trajectory in the x_1-x_2 plane. As time evolves the state of the system moves along this trajectory. Such a space is called a *phase space.*

In general $n>2$, and the *phase space* for an autonomous system (*i.e.* one not involving the independent variable, time, explicitly) is formed by the Cartesian product of the x_i's domains. Usually this consists of position momentum coordinates; however any linearly independent set made from the x_i will do. Moreover, the phase space is most commonly a plane although there is no difficulty in constructing more elaborate topologies such as the surface of a cylinder or a torus. We will assume a plane for simplicity.

A *representative point* in the phase plane specifies the instantaneous state of the associated system. As time evolves this point will move, shadowing the behaviour of the system. For systems which respond continuously this will be a continuous *trajectory*. However, if the system responds discontinuously the representative point will hop about from place to place. In some cases the sequence of iterates may eventually fall on a well defined curve in an orderly manner or, if the motion is chaotic, simply fill a region of the phase space with a splattering of points or fall along a curve in an apparently haphazard order (a strange-attractor). The systems of Lee & Kuipers, Struss and Sacks discussed below deal with the simpler continuous case whilst that of Yip, being numerical in nature, can accommodate more complex phase spaces.

The trajectories defined in this way fall into a finite number of categories, each one having a correspondingly unique qualitative behaviour. Closed curves (around a *vortex point*) indicate periodic motion; converging spirals (towards a *focal point*) indicate damped motion; diverging spirals attest to increasing amplitude oscillation; a trajectory approaching a (*nodal*) point in a well defined direction (*i.e.* not spiralling into it) indicates over-damped motion; hyperbolae (in the vicinity of a *saddle point*) suggest aperiodic motion; a trajectory separating two regions of dissimilar behaviour is known as a *separatrix*. Sometimes trajectories spiral inwards or outwards towards a closed curve known as a *limit cycle* whereupon the motion becomes stable periodic.

Certain configurations of trajectories are disallowed. For example, a trajectory may not cross itself: a corollary of this is that if we can prove a a trajectory passes through a point previously passed through the motion is periodic.

Various researchers have attempted to exploit phase plane analysis [Sacks 87b, Sacks 88, Yip 87, Yip 88, Struss 88e, Lee & Kuipers 88] although each has adopted a rather different approach.

The diagram below [Sacks 88 p.56] shows a portion of the phase plane for the Lienard equation [Sacks 88 p.16]

$$d^2x/dt^2 + dx/dt + x^2 + x = 0$$

and illustrates a number of features of phase portraits *e.g.* the inward spiral attests to damped oscillation.

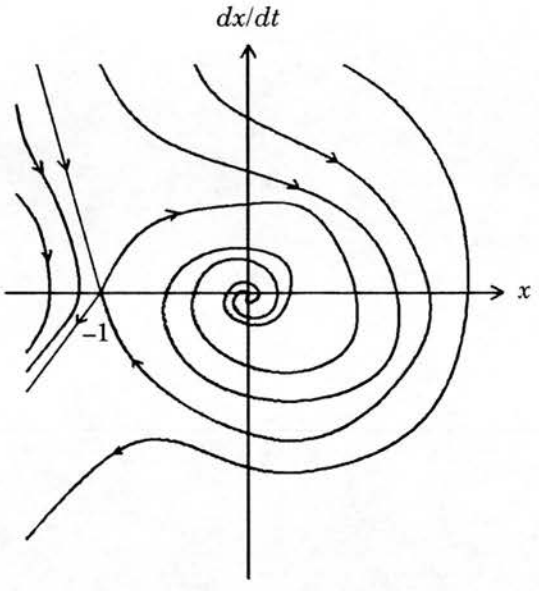


Figure 2-4. Phase Portrait of the Lienard Equation

2.4.3 Qualitative Phase Spaces

Lee & Kuipers' approach [Lee & Kuipers 88] begins by mapping a second order differential equation to a system of first order equations (this is trivially done by introducing an extra parameter to stand for the first derivative of the dependent variable, x). They then construct a phase space (with axes dx/dt versus x) and place points (x, dx/dt), representing pairs of qualitative values, on the phase plane. The dashed lines and axes represent landmark values in the quantity spaces of x or dx/dt. These define boxes in the phase space through which trajectories pass.

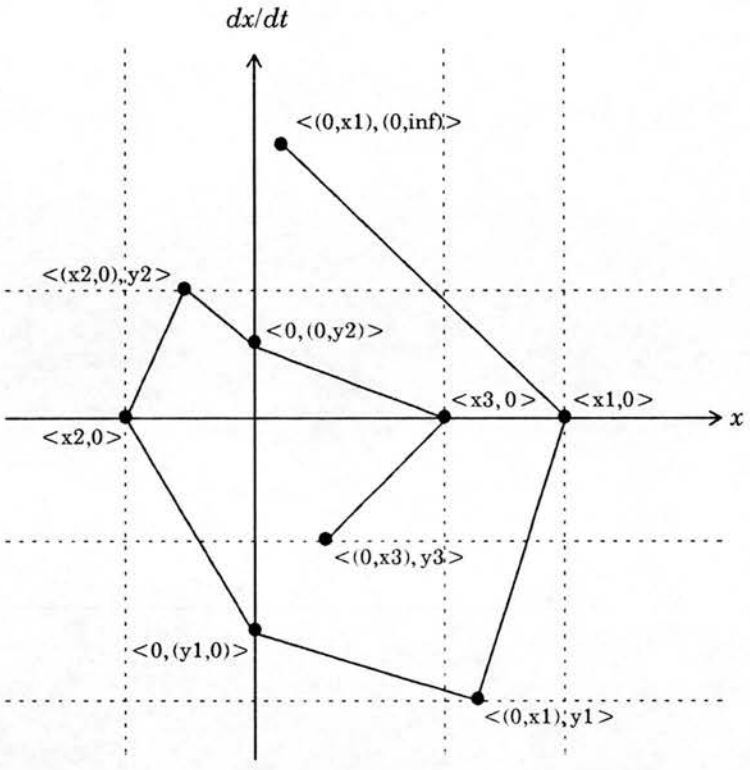


Figure 2-5. QSIM Phase Space of Damped Oscillator

The spurious behaviours which arise from direct qualitative simulation of the original second order equation can be identified as such using the "non-intersection constraint" in the phase plane. This prohibits trajectories from crossing by noting the edge along which a trajectory exits and then enforcing subsequent circuits through the box to be consistent with this. Thus, for the damped oscillator

$$dx/dt = v \ \wedge \ m\, dv/dt = -kx - \eta v$$

the phase plane approach avoids the non-physical solution of erratically varying amplitude oscillations predicted using the original **QSIM**. However, this system can still only deal with differential equations whose coefficients are not functions of the independent variable (time). Therefore systems such as forced oscillators cannot be modelled.

Struss

Struss' system [Struss 88e] is, in spirit, very similar to this. He again uses a version of the non-intersection constraint to identify spurious behaviours. The inferential apparatus

consists of rules which manipulate relations over behaviours (*i.e.* sequences of qualitative states) and single qualitative states. For example, "exclusive($bhvr_1$, $bhvr_2$)" means there is no system which admits both behaviours $bhvr_1$ and $bhvr_2$ and "convergent($bhvr_1$, $bhvr_2$, *qual-state*)" means the state sequences $bhvr_1$ and $bhvr_2$ both contain *qual-state* but have dissimilar immediate predecessor states to it. Hence at *qual-state* the behaviours "converge".

The rules encode logical consequences of conjunctions of these relations *e.g.*

$$\text{crossing}(bhvr_1, bhvr_2) \rightarrow \text{exclusive}(bhvr_1, bhvr_2)$$

(behaviours which cross are exclusive) and

$$\text{exclusive}(bhvr, bhvr) \rightarrow \text{spurious}(bhvr)$$

(a behaviour crossing itself is spurious). More complex rules permit reasoning about the symmetry of a phase space trajectory and inferences to the effect that a pair of behaviours are not exclusive. The latter is useful for inferring when two behaviours can legitimately be merged.

Although Struss provides the necessary language for reasoning about properties of the phase plane he fails to explain the meta-level control of his rule set. In fact the sample proofs he cites (for low dimensional systems) appear to have been hand-crafted. Consequently, one is left wondering whether this approach is truly general or contrived for a particular example. Moreover, although there has recently been considerable enthusiasm in the qualitative reasoning community for reasoning in the (qualitative) phase plane, only very simple systems have been tackled and it is not at all clear that the ideas generalise to more complex systems with higher dimensional phase spaces.

2.4.4 Numerical Integration + Qualitative Abstraction

A very different strategy is to integrate a system of (ordinary) differential equations numerically to construct a phase portrait. Such portraits can be analysed to suggest where

qualitatively interesting regions might lie *e.g.* by suggesting a possibly missing chain of *islands* or missing *separatrix etc.* Having done this, the numerical parameters are adjusted and the integration run again to generate some more detail in these regions. It is possible to distinguish six primitive types of long time behaviour and the modes of transition between them.

Advantages

- Tackles hard problems.
- Will scale up well.
- No spurious behaviours.

Disadvantages

- Requires a precise mathematical description of the system *i.e.* cannot accommodate non-numeric constants.
- Qualitatively interesting behaviour could be missed because of inappropriate choice of parameters values.
- A corollary: possibly faulty generalisation. The system may not behave a certain way for all numerical values in an interval simply because it does for some.

System: KAM

Yip adopts this approach [Yip87, Yip88]. In such a scheme four questions must be addressed:

1) where to start?
2) how to extract a qualitative behavioural description from purely numeric data?
3) how to control the selection of new parameter values for the next numerical experiment?
4) when to stop?

Failure to answer these questions condemns the reasoners to wallow in enormous search spaces. Yip summarizes the qualitatively distinct types of orbit which typify the class of equations he addresses: *periodic* orbit (closed curves), *almost periodic* orbit (iterates move around a well defined curve but never visit the same point twice), *island chain* (an almost periodic orbit surrounding a stable periodic orbit), *separatrix*, *chaotic* orbit (random splatter) and *escape to infinity* orbit. Some of these orbits cannot legitimately be adjacent in the phase space. **KAM** has rules delimiting the possibilities.

Orbit recognition is achieved using techniques borrowed from computational vision. Exploration begins in the vicinity of *stable fixed points* (a point at the centroid of a closed orbit) and extends radially outward until *island chains* and subsequently a *chaotic* regime is reached.

KAM constrains the search by identifying neighbouring orbits which violate some adjacency rule. These inconsistencies are maintained in a stack. While the stack is non-empty, **KAM** pops the first inconsistency and performs a bisection search of the region between the offending neighbours. This will reveal a new orbit type and lead to an update of the orbit adjacency record. Old inconsistencies are removed and new ones added in this fashion until the stack is empty at which point **KAM** terminates.

Review

Yip was the first to adopt a geometric, rather than state transition based, description of behaviour. The emphasis is on the topology of orbits in phase space rather than states and transitions and is based on solid mathematical foundations. Conversely, many qualitative calculi have recently found to be flawed [Struss 87, 88a, 88b, 88c].

KAM cannot accommodate partially specified systems (involving weak monotonic functional constraints) or even deal with ones containing non-numeric constants (as **PLR** can). Nor can the current system navigate usefully in chaotic regions. For example Yip states that it is difficult to find island chains embedded inside chaotic regions. It may also possible to dupe **KAM** into endless effort on certain classes of problem for which the phase portraits consist of self-similar trajectories as there would then be qualitatively interesting behaviour on every scale. However, Yip does not mention this possibility.

Notwithstanding these reservations, Yip's system is impressive and has the potential to become a very useful tool for assisting in the analysis of nonlinear systems of significant complexity.

2.4.5 Analytic Abstraction + Symbolic Integration + Qualitative Description

Another strategy, exploiting the geometric description, is to solve an analytic abstraction of the original problem and construct a qualitative description of the solution to the approximated equation *i.e.* a sketch (rather than a plot) of the phase plane.

Advantages

- This strategy inherits most of the advantages of the numerical scheme above but the validity of the final solution is sensitive to the validity of the approximations made initially.

Disadvantages

- The final solution, from which the phase portrait is drawn, is not a single function but a set of linear approximations over abutting intervals.

System: PLR

Sacks' **PLR** system [Sacks 87b, Sacks 88], falls into this category. **PLR** constructs a qualitative description of the phase space trajectories of the solution of a system of first order differential equations. Sacks tackles the general problem of a system of nonlinear equations by piecewise linearising their coefficients. This maps the nonlinear system to a set of linear systems each valid for some local region of the phase space. The separate linear systems can be solved using Sacks' previous **QMR** system [Sacks 85a, 85b] and the results concatenated to form a global analysis over the entire phase plane. The transition graphs predicted on the basis of different piecewise linearisations of the original equation are compared (indirectly because each transition graph defines regions in the phase plane

peculiar to the piecewise linearisation it came from). If they differ a finer piecewise linearisation is made. If they do not the program terminates.

PLR is a very impressive piece of work as it tackles hard problems. However, although the papers include diagrams presenting the trajectories graphically the implemented system produces, without user intervention, a transition graph rather than a phase portrait. To obtain the trajectory sketches the user must instruct the qualitative sketcher to draw the axes and the boundaries between regions where the qualitative behaviour is different. Then by clicking on one of these regions the system performs a numerical integration (Runge-Kutta order 4) of either the piecewise linearised or raw equations taking the selected point as initial condition. To do this the the system must assume numerical values for its symbolic constants. Sacks suggests that

- we can assess the veracity of the piecewise linear solution by visual comparison with either numerical integration
- we can assess how closely the piecewise linearised system approximates the true system by integrating both systems numerically and seeing how far trajectories which start at the same initial value diverge in the respective numerical integrations

So in order to assess the accuracy of **PLR** predictions the user is obliged to perform multiple numerical integrations for different initial conditions. This is not intended as a criticism of Sacks work: we mention it to highlight the fact that although the qualitative sketches in the papers look impressive and provide immediate insight into the qualitative properties of the solution, they are obtained at additional, and substantial, computational expense. What **PLR** gains, over Yip's approach is a guarantee that we have not skipped any qualitatively distinct mode of behaviour.

2.5 Merits of Approximate versus Qualitative Solutions

2.5.1 More Powerful Comparative Analysis

Weld's comparative analysis technique, DQ analysis, [Weld 87, 88b, 88c] allows the relative difference between the behaviours of two systems to be determined. However, Weld himself [Weld 87 p964] cites the following three ways in which DQ analysis can fail:

- *No Answer Possible* e.g. the query, "What would happen to the period of oscillation if the mass of the block was heavier and the spring was more stiff?" cannot be answered using qualitative information alone. The problem is that the pair of modifications introduce antagonistic effects and the qualitative calculus is too weak to resolve their combination unambiguously.

- *Incomplete Answer* (Weld calls this "*Ambiguous*") e.g. without including additional (conservation of energy) constraints DQ analysis cannot predict the overall effect on the period of increasing the mass of the spring-mass system. All it can do is determine that the mass takes longer to reach the first transition.

- *No Useful Perspective* e.g. the query "What would happen to the period of oscillation if the initial displacement is increased" is provably unanswerable using DQ analysis as there is no useful perspective for any parameter to allow a comparison to be made.

Approximate functional reasoning conquers all these deficiencies. As approximate functional reasoning deals with explicit, albeit approximate, mathematical functions the relative magnitude orderings of parameters to resolve antagonistic influences one way or the other can be determined. Second, approximate functional reasoning reveals global behavioural descriptions. Hence, there is no difficulty in determining the effect on an entire behaviour instead of simply a part thereof. Finally, functional knowledge can be used to answer queries which cannot be resolved using qualitative knowledge alone.

To illustrate the last point, consider a simple oscillator defined using the qualitative differential equation

$$d^2x/dt^2 = -\mathrm{M}^+(x).$$

Weld explains that, as there is no useful perspective, comparative analysis cannot give an answer to the question "What would happen to the period of oscillation if the initial displacement is increased". For a small amplitude linear pendulum the correct answer is "period is unchanged". This is the answer Weld would like his system to return. However, this is only *necessarily* true of a *linear* oscillator. The qualitative differential equation above is in fact that of a family of oscillators, including nonlinear oscillators. The problem is that if the above qualitative differential equation is used as a basis for simulating the behaviour of the associated system there is no way of knowing whether $\mathrm{M}^+(x)$ corresponds to a linear or nonlinear function. Hence **DQ** analysis is not only unable to say what will happen if the initial amplitude is increased because of the lack of a suitable perspective but, more seriously, because an unambiguous answer is theoretically impossible without functional knowledge embodied in $\mathrm{M}^+(x)$.

Provided the original ordinary differential equation is known, the technique of **analytic abduction**, which we introduce in *Chapter 3*, not only has the potential to answer this question but, more importantly, to answer it in different ways depending on whether the restoring force is linear or nonlinear. If Weld were to abstract the restoring term to a monotonic constraint, as above, then the essential difference between the linear and nonlinear oscillators would be lost.

Weld's second comparative analysis technique, **Exaggeration** [Weld 88a], is also capable of returning erroneous answers because it implicitly assumes the system responds monotonically to a perturbation. Moreover, the branching factor in **HR-QSIM** is even higher than in **QSIM** so even moderately complex systems would be difficult to simulate. **B**ack **O**f **TH**e **E**nvelope **R**easoning, which we introduce in *Chapter 8*, is the approximate functional equivalent of **Exaggeration**. However, as **BOTHER** deals with explicit functions instead of and arbitrary limits, it is capable of making more precise inferences by returning the functional behaviour of some expression at a magnitude extreme. However, this is to be

expected given the richer representation embodied in functional rather than qualitative expressions.

The case for qualitative comparative analysis is that it can sometimes (Weld might say often) answer perturbation type queries pertaining to partially specified systems such as the weak qualitative version of the oscillator equation above. However, if functional knowledge is available, approximate functional reasoning offers an interesting alternative to purely qualitative reasoning without incurring the complexity of exact analytic techniques.

2.5.2 Behavioural Compression

Another way of viewing approximate functional solutions of differential equations is as a means of compressing a behavioural prediction to a comprehensible form. Previous analytic or numerical equation solvers (**QMR**. **PLR**, **KAM**) make no attempt to do this: either the solution is presented verbatim else the trajectories it engenders in the phase plane serve as the highest level behavioural description.

For systems which reason with qualitative differential equations [Kuipers 85, Kuipers 86a, de Kleer & Brown 84, Weld 88c, Weld 88d], work on behavioural compression techniques has focused on recognising cycles in process histories [Weld 85, 86] or the suppression of irrelevant differences [Kuipers 87, Kuipers & Chiu 87].

For example, if two parameters are changing in the same direction their difference can be positive, zero or negative. Behaviours can be predicted which are for the most part identical but differ by the sign of this difference. Such ambiguities can cause a combinatorial explosion in an envisionment. One way to resolve the problem is to adopt a "don't care" approach and to aggregate the three possibilities to a new qualitative value "*ign*" (for *ignore*) [Kuipers & Chiu 87]. In the **QSIM** approach the only complication is that we need to be careful when determining successors to states containing *ign* that there is at least one consistent labelling of *ign* with *inc*, *std* or *dec* which would satisfy the proposed transition. If, for example, the only consistent post-transition state were $<0, inc>$ and its predecessors

were either $<+,dec>$ or $<+,std>$ and we represented this as the transition $<+,ign> \rightarrow <0, ign>$ then it would not be valid as such a transition would violate continuous differentiability. Behaviour merging is inefficient because it expends effort on generating behaviours only to combine them later.

If the timescales of various processes in a mechanism are known, compressed behaviours may be derived directly by modifying the structural description [Kuipers 87]. These are "compressed" in the sense that extraneous temporal information is avoided by the simulator. The idea is to decompose a complex system into a set of interacting equilibrium mechanisms which operate over vastly different timescales. This permits the structural description to be modified depending on perspective: a slow mechanism can replace complex relations between itself and a faster mechanism as functions returning instantaneous values. A fast mechanism can replace parameters which vary by a slower mechanism as constants.

A rather different approach, *aggregation* [Weld 85, Weld 86], is a technique for recognising cycles, and cycles within cycles, in a sequence of processes. First a *repetition recognizer* examines the set of currently active processes to see if any are the same as one previously occurring in the process history. Since the action of processes might well modify some object, the notion of "sameness" must be sufficiently flexible to accommodate unessential differences in the process instance. Second, a *cycle extractor* combs through the history to determine the sequence of activity actually involved in reactivating the process. This is not necessarily an unmodified trace of the intervening processes as certain of their effects may be quite irrelevant whilst others crucial. Finally, a *cycle verifier* checks that the proposed cycle can actually repeat either by simulating it through two further iterations or testing for equality of values (which is much stronger than "sameness") in preconditions. Neither technique is always applicable as complications can arise from interference of other processes or ambiguities in process activation orderings. The final aggregated cycle has preconditions and influences derived from the processes subsumed. Limit analysis may then be applied to determine the cycle's ultimate state.

By contrast the approximate functional level yields a compiled behavioural description rather than the sequences of states generated by **QSIM** or the graph of possible state to state transitions in envisionments.

Moreover, if **QMR** or some other symbolic manipulation system solved a differential equation exactly the approximate functional description would generally be more compact than this too. This is because **BOTHER** would work an expression hard before returning it to the user.

2.5.3 Absolute Timescale

Qualitative behaviour predictions [Bobrow 84] lack a measure of absolute time: their time points are merely ordered without scale [Kuipers 86a, Kuipers 87, Williams 86, Allen 83]. Approximate functional solutions of time dependent phenomena mention time explicitly and may therefore support temporal queries unanswerable by qualitative reasoners.

2.6 Conclusions

This chapter has reviewed the *approximate functional*, the *qualitative* and the *geometric* paradigms for reasoning about the solutions of differential equations. Our principal objective was to convince the reader of the legitimacy and utility of approximate functional solutions and to emphasise their advantages, in certain circumstances, over other types of representation. Previous work in artificial intelligence has concentrated on the latter two paradigms. However, if a system can be specified using an ordinary (rather than qualitative) differential equation, then the approximate functional level has the potential for yielding much more specific inferences than those typically produced by qualitative reasoning alone. Moreover, approximate functional reasoning supports more sophisticated comparative analysis queries, compresses behaviours more concisely and has a richer temporal representation than qualitative reasoning. Finally, it is superior to exact methods by sacrificing accuracy in favour of concise yet adequate solutions which are more easily comprehended and emphasise the coarse functional relationships of the true solution.

Approximate functional reasoning is not, however, a panacea. Nor, for that matter, are any of the approaches reported in this chapter. Each is tailored to reasoning with various levels

of specificity in the description of a dynamic system ranging from *numeric* through *geometric, exact analytic, approximate functional, order of magnitude,* and *qualitative* descriptions. As implied by this ordering we view approximate functional reasoning as occupying a niche somewhere between exact analytic solutions and weaker qualitative varieties.

Sacks [Sacks 85a p139] shares our view of the importance of the two key issues addressed in this thesis: *unwieldy* and *open form* solutions. Exact solutions are of little value if they are so complex as to be incomprehensible. However his approach to the problem has been to piecewise linearise nonlinear equations and construct their phase portraits. The emphasis is thereby shifted to the properties of phase plane trajectories and their decoding as corresponding qualitative behaviours but the underlying equations driving the phase portrait may still be very complicated. Our approach is to cut through mathematical details and attempt to reason with coarse, but adequate, approximations.

The rest of this thesis develops the tools necessary to reason at the approximate functional level. *Chapter 3* introduces **Analytic Abduction** which implements a weighted residual method. We chose collocation as this circumvents the need for explicit integration. One new feature is the way qualitative reasoning is integrated with analytic reasoning to control the selection of suitable trial functions. A second is the way we navigate through the space of possible trial functions to pick a better one if our first choice is found wanting.

Chapter 4 introduces **Segment Calculus** for reasoning about the qualitative interaction of curve segments under certain mathematical operations. This will allow us to identify compound functions which could possibly account for a qualitative behaviour but which cannot be directly inferred by analytic abduction.

At the very least **Segment Calculus** can suggest a possible *form* for a solution and we can use this knowledge in **Equation Parsing** developed in *Chapter 5* to attempt to find a true closed form solution.

Chapter 7 presents **Closed Form Approximation**: a method for implementing series solution (described in *Chapter 6*). **Closed form approximation** solves regular singular differential equations exactly, but in open form, and then approximates the infinite series

solution in closed form. A novel feature of this procedure is our representation of infinite series.

Chapter 8 explains our **Back of the Envelope Reasoner** (BOTHER) for performing very aggressive approximate functional inferences in arbitrary limits.

Finally *Chapter 9* compares these techniques showing how one can succeed where another fails. Some of the problems we address may be solved using other, perhaps even exact, techniques. Our interest in the approximate functional level stems from the hope that the ideas we develop in simple systems will carry over to more complex problems in the future. As the approximate functional reasoning is based on well respected techniques from engineering this seems plausible.

Chapter 3

Analytic Abduction

3.1 Introduction

In this chapter we describe **analytic abduction**, a technique for finding an approximate solution to an ordinary differential equation. The technique consists of envisioning the qualitative properties of the solution and then abducing a functional form which is both *congruent* to the qualitative behavioural description and *adequate* with respect to the underlying differential equation. *Congruence* captures the intuition that the trial function has the right shape and *adequacy* that it satisfies the differential equation sufficiently well.

Analytic abduction is akin to finding an approximate solution by a weighted residual minimization procedure. However, it differs from previous work [Crandall 56, Finlayson 72, Acton & Squire 85] in two respects: first we are interested in finding a function which gives a reasonable approximation using only one term. Previous work has focused on finding approximations in terms of a series of orthogonal functions. Our objection to this is that such series may not easily be comprehended *i.e.* understood qualitatively. By using single term approximations we hope to find succinct approximate behavioural descriptions. Second, we use the failure of one trial function to guide the search for a better

approximation. The Acton & Squire text ignores the problem of what to do when the first trial function is found to be a poor approximation. Other authors suggest adding in another term from the family of orthogonal functions in the trial series. Our approach introduces three methods of repair. The first two, called *topological isomorphism* and *topological similarity,* identify a qualitatively similar, but mathematically distinct, type of function which is hoped to yield a better approximation. The second, called *exaggeration,* is invoked only if the other methods fail and works by sacrificing accuracy at the initial state in favour of a better fit overall.

Approximate solutions are better than qualitative solutions because they define a *time scale* for the behaviour, summarize the *functional* relationships between parameters and support more sophisticated *comparative analysis* queries. Moreover, they are better than exact mathematical solutions by being simpler to comprehend and manipulate. For example, the exact solution of

$$dn_c/dt = K(n_a - n_c)(n_b - 2*n_c)^2$$

is

$$t = (1/K)\,[(1/(2n_a - n_b))(1/(n_b - 2n_c) - 1/n_b) + (2/(2n_a - n_b)^2)\log_e[(1-2n_c/n_b)/(1-n_c/n_a)]\,]$$

[Acton & Squire 85 p2]. However, it is both hard to comprehend the solution and understand the effect of adjusting n_a or n_b in qualitative terms. Conversely, the approximate functional solution of this equation is of the form

$$n_c = n_\infty(1-\exp(-[0.63\,(2Kn_an_b)^{0.36}]\,t^{0.36}))$$

(an improvement on Acton & Squire's solution) which we derive in §3.6.3. From this it is much easier to determine the qualitative effects of varying n_a and n_b. The **analytic abduction** algorithm can solve this problem.

3.1.1 Query Types

A major advantage of reasoning at the approximate functional level is the range of queries it allows to be answered. These include

- obtaining the (approximate) functional relationships governing some behaviour
- obtaining an absolute scale *e.g.* by estimating a time or length constant
- estimating the time for a process to complete
- predicting the effect on behaviour of changes to certain parameters (by deducing the relative magnitude orderings under which different behaviours arise)

The last point is subtle. Weld's comparative analysis [Weld 88c] answers some kinds of query but is stumped if a variation induces antagonistic effects *e.g.* if $y = x + 1/x$ what happens to y as $x \uparrow$. This is unanswerable qualitatively as $x \uparrow$ and $1/x \downarrow$. However, the approximate functional level is much richer. Since we have *functional* knowledge, albeit approximate, explicitly encoded, we can differentiate such an expression with respect to x to infer the effect unambiguously.

3.1.2 Cognitive Model

Our model of the reasoning process underlying **analytic abduction** is abstracted from the examples found in Acton & Squire [Acton & Squire85]. This text collates various tricks-of-the-trade based on years of teaching mathematics to students who needed to apply it in the real world. In a sense they have done the knowledge elicitation for us. However, their account lacks sufficient detail to automate the procedure because they simply assume that the qualitative form of the solution to an equation can be obtained "by commonsense" and say nothing about how to recover from failure and come up with a better guess.

However, this is not intended to decry their contribution. Indeed we can deflect many criticisms relating to the utility of approximate solutions by referring the critic to this text.

3.1.3 Plan

The rest of the chapter is organised as follows: §3.2 presents an overview of the modified weighted residual procedure we call **analytic abduction**. Section §3.3 exposes the issues that must be addressed to permit automation of this procedure. These are dealt with in §3.4 where we present the **analytic abduction** algorithm in full. §3.5 then gives some examples of **analytic abduction** in action including both successes and failures. Some ways of recovering from failures are developed in §3.6. §3.7 focuses on the question of what knowledge should be explicitly encoded and §3.8 describes the current limitations of the system. Finally we summarize the research contributions of this chapter and link **analytic abduction** to other techniques found in the thesis.

3.2 Overview of Analytic Abduction

Analytic abduction comprises four independent techniques: *qualitative simulation, trial function selection, parameter optimisation* and *verification*. The first two allow a parameterised closed form function to be determined which is topologically equivalent to the qualitative behaviour and the third to optimise it with respect to the original differential equation by adjusting its parameters. Finally the approximations so computed are checked to ensure that they have not strayed outside permissible bounds. If they have, a *topologically isomorphic* or *topologically similar* trial function is selected and the optimisation and verification stages are repeated. If this too fails, the problem constraints are weakened by *exaggerating* the initial condition and the **analytic abduction** procedure is recursed on again. This cycle is summarized in Figure 3-1.

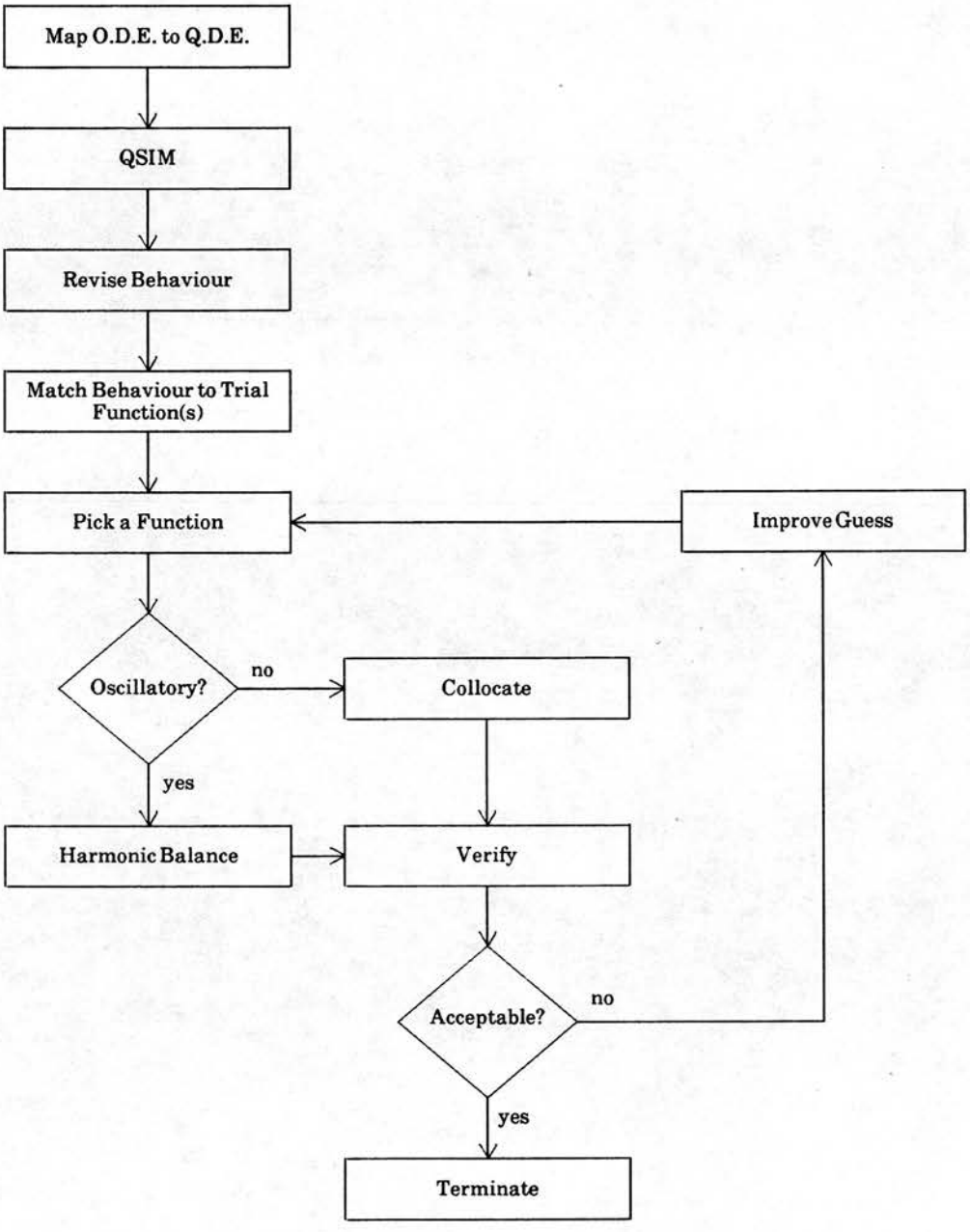


Figure 3-1. The Analytic Abduction Algorithm

3.3 Issues

This high level description of **analytic abduction** raises a number of interesting issues as regards the automation of the procedure.

1) How can the qualitative properties of the solution be deduced automatically?

2) How are analytic functions to be represented so as to facilitate matching them to qualitative behaviours?

3) How can a good initial candidate function be selected given competing topologically equivalent alternatives?

4) How is the *adequacy* of the approximation to be assessed?

5) Having found a candidate to be wanting is it possible to improve upon it?

6) By what criteria should the "standard" functions be selected? Traditional weighted residual minimization methods adopt basis functions *e.g.* trigonometric functions, Legendre, Hermite, Laguerre, Chebeyshev polynomials. We want to leap to a good approximation in one step so these functions are not necessarily the best.

7) How are the collocation points chosen (many standard methods use the roots of whatever polynomials they take as their basis functions).

These issues are addressed in the subsequent sections as we describe the stages of **analytic abduction**.

3.4 Analytic Abduction

3.4.1 Qualitative Simulation

The first step in **analytic abduction** uses a reconstruction of **QSIM** to envision the qualitative behaviour of the solution. From the original differential equation and an initial state description, **QSIM** generates a tree of possible qualitative states. Each sequence of states on a path from root to leaves represents a possible qualitative behaviour. A single qualitative state contains information about the value of parameters in the system in relation to a set of landmark values together with the signs of their derivatives. Landmark values are critical or boundary values of the dependent variable. The novel feature of **QSIM** in comparison to other qualitative reasoning systems is that it can dynamically discover new landmark values as the simulation proceeds. This is done by generating a nominal symbol (**LMRK1**, **LMRK2**, **LMRK3** *etc*) whenever the derivative of some parameter becomes zero and inserting it in between the appropriate existing landmarks.

3.4.2 Trial Function Selection

3.4.2.1 Qualitative Behavioural Revision

QSIM generates behaviours as sequences of tuples of qualitative states. If functions are likewise represented we can define a function and a behaviour to be *congruent* if they possess the same sequence of critical points, convexities and bounding intervals. Raw output from our version of **QSIM**, however, is not immediately useful for this comparison. This is because as simulation proceeds new landmark values may be discovered and inserted between existing ones. Therefore, the landmark sets at the beginning and end of a simulation may differ. Hence the output is revised into a globally consistent qualitative behaviour by propagating the discovery of the new landmark values back to the beginning of the simulation. After such a revision, the initial and final landmark sets will be the same and some qualitative value intervals will be shrunk to accommodate the new knowledge. Revision cannot take place whenever a new landmark is discovered because qualitative behaviours sharing common beginnings have common stems in the tree of possible

behaviours, yet, for nodes at any given level, the landmarks beneath one are independent of those beneath another. Consequently it would wrong to revise back up to the root as soon as a new landmark is discovered.

Figure 3-2 shows the effect of revision on the qualitative behavioural description of a simple constant amplitude oscillation between I and LMRK2 before and after revision (initially the critical point LMRK2 was unknown to the system). Throughout this chapter $f(t;b)$ will represent a function of time parameterised by b.

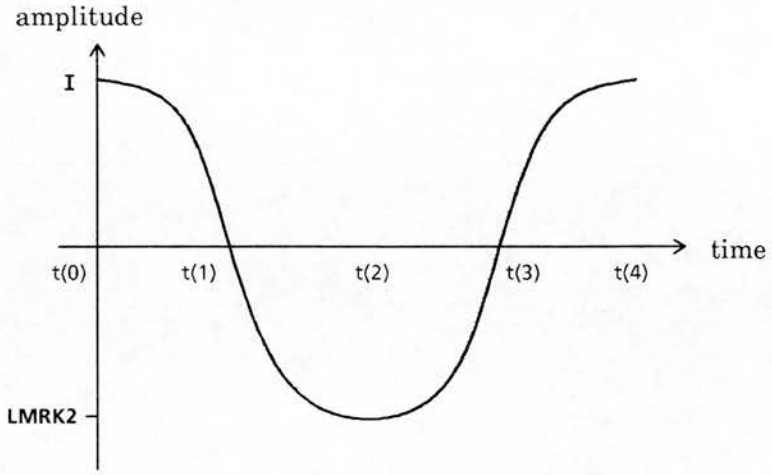


Figure 3-2. Constant Amplitude Oscillation

	Before Revision		After Revision	
time	$f(t;b)$	$df(t;b)/dt$	$f(t;b)$	$df(t;b)/dt$
t(0)	<I,std>	<0,dec>	<I,std>	<0,dec>
t(0,1)	<[0,I],dec>	<[-inf,0],dec>	<[0,I],dec>	<[LMRK1,0],dec>
t(1)	<0,dec>	<LMRK1,std>	<0,dec>	<LMRK1,std>
t(1,2)	<[-inf,0],dec>	<[LMRK1,0],inc>	<[LMRK2,0],dec>	<[LMRK1,0],inc>
t(2)	<LMRK2,std>	<0,inc>	<LMRK2,std>	<0,inc>
t(2,3)	<[LMRK2,0],inc>	<[0,+inf],inc>	<[LMRK2,0],inc>	<[0,LMRK3],inc>
t(3)	<0,inc>	<LMRK3,std>	<0,inc>	<LMRK3,std>
t(3,4)	<[0,I],inc>	<[0,LMRK3],dec>	<[0,I],inc>	<[0,LMRK3],dec>
t(4)	<I,std>	<0,dec>	<I,std>	<0,dec>

By representing the qualitative properties of known functions and their first and second derivatives as $<qval,qdir>$ pairs as above we can match the qualitative behavioural prediction of **QSIM** to templates corresponding to known analytic functions.

3.4.2.2 Known Functions

Given a revised qualitative behavioural prediction the task is to identify parameterised functions which could possibly account for the observed behaviour. To do this it is necessary to have an explicit representation of the qualitative properties of known functions.

Without functional information, the sign of the zero-th and every higher derivative is required to accurately specify a behaviour [Morgan 88]. To make a practical system it is therefore necessary to truncate this sequence of signs at some point. A convenient compromise is to represent just the $<qval,qdir>$ pairs of the function and its first derivative. It seems implausible that humans base their abductions on higher derivative information. Each function is therefore represented by an object whose attributes include

name:	the conjecture's name
function:	the defining functional form
descriptor:	the sequence of pairs of qualitative states of the function and its first derivative
consequences:	a set of relations and predicates this conjecture entails
parameters:	the adjustable parameters in the conjecture

For efficiency, other information *e.g.* derivatives and the inverse function are also stored.

We cite some simple examples below. More complex ones, *e.g.* including a horizontal shift, are easily represented in a similar way. As an illustration, a rising exponential function (see overleaf)

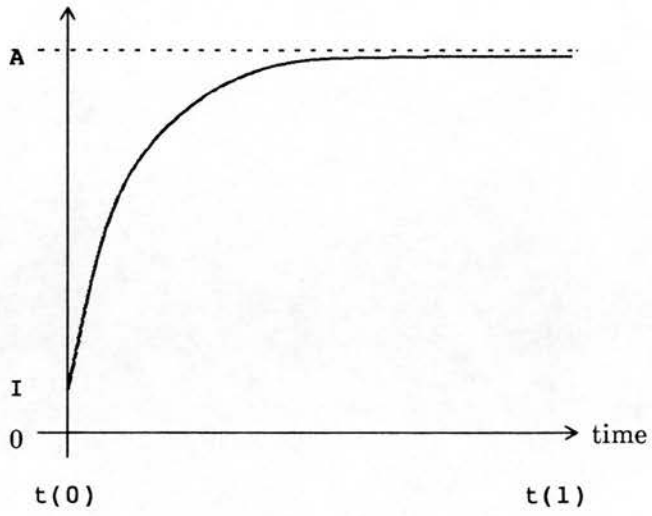


Figure 3-3. Rising Exponential

would have the following function object:-

name: rising-exponential

function: `F = A - (A-I)*exp( - b*t)`

descriptor:
```
{t(0):   [<I,inc>,      <[0,+inf],dec>],
 t(0,1): [<[I,A],inc>, <[0,+inf],dec>],
 t(1):   [<A,std>,      <0,std>]}
```

consequences: {**A>I**,
asymptotic-approach(**F**,**A**),
horizontal-shift(**0**),
vertical-shift(**I**),
vertical-scale(**A**),
is-infinite(t(1))}

parameters: [b]

3.4.2.3 Matching Qualitative Behaviours to Function Descriptors

We define a function, **F**, to be *congruent* to a qualitative behaviour if its descriptor entry has the same sequence of critical points, convexities and bounding intervals. This is easy to ascertain simply by attempting to unify the descriptor template with the revised qualitative

behaviour. In Prolog this amounts to a one line sub-goal, "`FDescrip = RevisedQBhvr`". If the goal succeeds, `F` becomes a trial function.

Trial function conjectures may not be unique since many different functions share the same qualitative description. The philosophy we adopt is to randomly pick a function congruent to the revised qualitative behaviour, examine its suitability and if it fails use the mode of failure to find a better approximation. In the first instance the system attempts to satisfy all the problem constraints but if this fails it will then proceed to relax the boundary constraints in an effort to find a better fit overall.

3.4.3 Parameter Optimisation

As the exact solution is not available (even as a table of numerical data) the trial function cannot be optimised directly (*e.g.* by a least squares procedure). Instead, the optimisation must be done indirectly *via* its "fit" with the original differential equation. This is accomplished by adjusting the values of the free parameters in the trial function until it agrees as closely as possible with the differential equation. This procedure is essentially a weighted residual minimization method discussed in the last chapter. The algorithm is abstracted from the hand-cranked examples in Acton & Squire with two novel features: automation of qualitative envisionment and an extension of the theory to recover from the failure of poor trial function conjectures.

Clearly if the trial function is not the true solution it cannot be made to satisfy the equation throughout the entire domain and will give rise to an error term, R_e, known as the equation residual. The goal of parameter optimisation is to find values for the adjustable parameter(s) in the trial function, b_i, such that the weighted average of the residual is zero. We have implemented two methods for doing this: Collocation (for non-oscillatory systems) and Harmonic Balance (for oscillatory systems). Other methods are possible [Crandall 56] but all involve the calculation of integrals and consequently have a higher computational overhead.

3.4.3.1 Collocation

Algorithm 3-1. Collocation

1) substitute the conjecture into the differential equation to form the residual equation

2) count the number of adjustable parameters b_i $(i=1,2,...,n)$

3) set R_e to be zero at each of n points (chosen according to the *coverage* and *curvature* heuristics below). This results in a system of n equations for n unknowns.

4) solve for the b_i

Rationale:

If the true solution is continuously differentiable (as it is in `QSIM`) and it is made to agree with the conjecture at n points then it will not stray too far from the conjecture in between these points (this is the implicit inference made when least squares fitting a smooth curve through a set of data points).

Which Points to Choose?

Recall from *Chapter 2* that the precise choice of collocation points is, in principle, arbitrary. In standard collocation, where a function is approximated with a finite series of orthogonal polynomials, the collocation points are often chosen to be the zeroes of the polynomials to simplify the calculations. However, in our case two things are different: first we are interested in single term approximations rather than series. This suggests better results might be expected if "representative points" are chosen *i.e.* points neither too close together nor too near to the boundaries and preferably in the vicinity of maximum change in the function. Second, the time axis is an unscaled sequence of points and intervals. So it is not possible to pick an absolute value along this axis. Instead we use *collocation fractions*, by which we mean a fraction of the range of the function. So whereas standard collocation (including the Acton & Squire text) choose points in the domain of the function, our system essentially chooses points in its range, represented as fractions of the difference between its

maximum and minimum (it is simple enough to identify the corresponding collocation point by inverting the function). If we suppose the range of the trial function is bounded between **min** and **max** we invent two heuristics to guide the choice of collocation fractions:

coverage heuristic:	the collocation fractions should be roughly evenly distributed over (**min, max**) or
curvature heuristic:	weighted towards the region of maximum curvature

Each collocation fraction determines a numerical approximation for the conjecture which allows its elimination from the residual equation. For example, *congruence* guarantees that the conjecture satisfies the initial and final states. A choice for collocation point under the *coverage* heuristic would then be half way through the change *i.e.* at (**min** + **max**)/2.

Equating the function to a collocation fraction allows a time, t_c, to be determined (by inverting the function) at which the equation residual is chosen to be made zero. In most cases it is not necessary to invert the function explicitly as the only time dependent term in the equation remains embedded in some functor. For example, the range of a rising exponential is bounded between I (the initial ordinate intercept) and A the final asymptotic value. To find the time corresponding to the half way stage, $t_{1/2}$, the function is equated to the value $(A + I)/2$ (collocation fraction, $c = 1/2$). This gives

$$A - (A - I)*\exp(-bt_{1/2}) = (A + I)/2$$

which implies $\exp(-bt_{1/2}) = 1/2$. Although it is possible to isolate $t_{1/2}$ by inverting the exponential, this will be unnecessary as derivatives of the exponential function (linear in t) do not generate t in isolation. Hence when this trial function is substituted into the original differential equation, which did not contain time dependent coefficients to begin with, the variable time will never appear outside the exponential. Other trial functions, however, such as an exponential quadratic in t, *i.e.* $\exp(-bt^2)$, *will* have to be inverted to find t explicitly as their derivatives do generate new terms in t. Conceptually it is best to imagine that t is always isolated.

The upshot of this procedure is that it is possible to find a value for t_c in terms of the landmark values of the qualitative simulation and known functions. This allows t_c to be eliminated from the equation residual and to determine a value for the adjustable parameters which will guarantee the residual is zero *at time* t_c. The hope is that the equation residual will not stray too far from zero in between the collocation points.

A similar argument applies for the *curvature* heuristic. For example, for the falling half parabola, with ordinate intercept I and ordinate extremum A, a choice weighted towards the region of maximum curvature would be $(I+8A)/9$:-

$$I+(A-I)[1-((t-b)/b)^2] = (I+8A)/9$$

which implies $((t-b)/b)^2 = 1/9$.

Although these simplifications are easily computed, *e.g.* using **PRESS** [Bundy & Welham 81, Bundy & Sterling 81, Sterling *et al* 82, Bundy 83] if t_c is easily isolatable or **BOTHER** (*Chapter 7*) if it requires approximation, it is more efficient to record them as collocation schemata which are invoked whenever the corresponding function is conjectured. This means that as soon as a trial function is hypothesised, the variables in it are automatically instantiated to the appropriate landmark symbols and its derivatives and substitutions for various collocation fractions are retrieved. Although this may seem rather rigid we must remember that the choice of collocation fractions is, theoretically speaking, arbitrary so there is little to be gained performing unnecessary symbolic manipulation at run time.

3.4.3.2 Harmonic Balance

For oscillatory systems we adopt a different approximation technique which is known, empirically, to give better results. The following algorithm is immediate from Acton & Squire's account [Acton & Squire 85 p83]. Unlike collocation, where we contribute to the theory (§3.6), we have not yet extended harmonic balance beyond their description of it. Nevertheless, the computer implementation of this algorithm is original.

Algorithm 3-2. Harmonic Balance

1) substitute the conjecture into the differential equation to form the residual equation

2) rewrite powers of sinusoids as multiple angle formulae

3) arrange the residual equation in terms of ascending multiple angles

4) count the number of adjustable parameters (b_i: $i = 1,2,...,n$)

5) equate the n lowest order terms to zero (assume any higher order terms are negligible in comparison to lower orders)

this results in a system of n equations in n unknowns

6) solve for the b_i

Rationale:

The residual equation approximates the start of a Fourier series of the solution. If this series converges sufficiently rapidly then we can approximate the true solution by finding those parameter values such that only the n lowest order terms are balanced.

3.4.4 Verification

Approximation techniques are only useful if it is possible to assess their veracity. Each of the techniques described above rested on some fundamental assumptions: for collocation it was that the parameter values were insensitive to the exact choice of collocation fraction, whilst for harmonic balance it was that the series whose beginning was represented by the residual equation was rapidly convergent. It is therefore possible to assess the accuracy of the approximate inferences by assessing the accuracy of these underlying assumptions.

3.4.4.1 Adequacy Criterion for Collocation

If the conjecture $f(t;b_i)$ happened to be the true solution, then the b_i would be the same regardless of the collocation fraction. However, in general this will not be the case. As a specific example, consider the rising exponential function $f(t;b) = 1\text{-}\exp(-bt)$. Suppose that two different collocation fractions predict different values for b, say a_1 and a_2 and that their ratio is n. Then

$$f(t;a_1)/f(t;a_2) = (1-\exp(-a_1t))/(1-\exp(-a_2t))$$

Now $\lim(t \to \infty)\, f(t;a_1)/f(t;a_2) = 1$, so at late times the choice of collocation fraction is irrelevant. However, $\lim(t \to 0)\, f(t;a_1)/f(t;a_2) = \lim(t \to 0)\, (1-\exp(-na_2t))/(1-\exp(-a_2t))$ $= n$ (by series expansion). So the error will be most keenly felt at early times. At worst, the value of the function they predict will also vary by a factor of n.

We allow the user to choose a maximum percentage error by which two supposedly equal parameters may differ, expressed as a fraction of 1. Call this p, then clearly we require

$$\max(|(a_1-a_2)/a_1|, |(a_1-a_2)/a_2|) < p$$

which by considering the cases $0 < a_1/a_2 < 1$ and $a_1/a_2 > 1$ implies

$$1/(1+p) < a_1/a_2 < 1+p$$

Therefore, a necessary (but not sufficient) criterion for testing the applicability of the predictions of collocation is that the ratio of the values of a parameter calculated at different collocation fractions should be bounded by the above inequalities. In all the examples presented in §3.5 we set p to be one.

3.4.4.2 Adequacy Criterion for Harmonic Balance

Harmonic Balance presupposes that the residual equation approximates the start of a rapidly convergent Fourier series. This assumption can be tested by examining the ratio of

the amplitude of the lowest neglected subterm to the amplitude of the restoring term in the original oscillator equation.

To obtain the numerator, recall that terms in the residual equation are arranged in order of ascending harmonics. If there are n adjustable parameters, harmonic balance utilises the first n terms. Therefore, the lowest neglected subterm will be the $(n+1)$th.

The denominator is found by comparing the original differential equation with the general equation of an unforced oscillator:

$$d^2u/dt^2 + P(u)\, du/dt + Q(u)\, u = 0$$

conceptually,

$$\{\text{acceleration}\} + \{\text{damping}\} + \{\text{restoring}\} = 0$$

so it is easy to identify the restoring term and hence both components of the ratio. In accordance with common engineering practice [Acton & Squire 85] if this ratio turns out to be less than $1/3$ (or 33%) harmonic balance is deemed to be reasonably accurate.

The techniques used for optimisation and applicability verification are based on standard engineering practice [Crandall 56, Finlayson 72] but we believe our system is the first to exploit them computationally and has necessitated formalising their more *ad hoc* elements.

We have not yet discussed how to recover from cases where our first trial function is inadequate. However, we postpone this until after some examples which demonstrate the successful application of the **analytic abduction** technique developed thus far.

3.4.5 Algorithm

In short, the **analytic abduction** procedure may be summarized in the following algorithm:

Algorithm 3-3. Analytic Abduction

Input

- an ordinary differential equations (O.D.E.)
- an initial state description

Method

- map the O.D.E. to a qualitative differential equation (Q.D.E.)
- simulate the the qualitative constraint system (using **QSIM**) to determine its behaviour
- propagate knowledge of any newly discovered landmarks back to the initial state description
- find a set of trial functions whose qualitative properties match those of the revised behaviour
- pick a trial function
- (*) if oscillatory
    - optimise its adjustable parameters *via* harmonic balance
- if non-oscillatory
    - optimise its adjustable parameters *via* collocation
- in either case verify the proposed solution is an *adequate* approximation with respect to the O.D.E.
- if it is, terminate
- if not, invoke *topological isomorphism*, *topological similarity* or *exaggeration* (in that order) to find an alternative trial function and recurse from (*)

3.5 Examples

The following examples appear in [Acton & Squire 85] as exemplars or unsolved end of chapter problems. Each is solved by applying the **analytic abduction** algorithm. The advantage of using examples from existing texts is that we can easily compare the output from **analytic abduction** against their hand-cranked counterparts. However, the steps in **analytic abduction** do not always mirror the text book derivations. This is because **analytic abduction** is a rationalisation of the approximation procedure so that the same algorithm applies to a wide class of problems and, moreover, because we have extended the theory to deal with cases where the first trial function fails. The Acton & Squire text is mute on such issues.

Below we present three successful applications of **analytic abduction** followed by an example for which the technique developed thus far fails. This suggests an extension to the theory is necessary and we pick this up again in § 3.6.

In order to improve legibility we have rewritten the equations occurring in the examples by hand. In the implementation all equations are normalised prior to manipulation. The normaliser is an extension of that used in **PRESS** whose major features are as follows:

- $-X$ is rewritten to $-1*X$
- $X-Y$ is rewritten to $X+-1*Y$
- X/Y is rewritten to $X*Y \uparrow -1$
- *rationals* & *integers* are rewritten as *reals*
- common variables are collected
- numerical subterms are evaluated

3.5.1 Collocation

Consider the equation of motion of a falling stone, $m*dv/dt = m*g - b*v^2$, where m is its mass, v its instantaneous velocity measured positive downwards, g the acceleration due to gravity and b a coefficient of resistance. Furthermore, air resistance is assumed to be proportional to the square of the stone's instantaneous velocity.

Map Differential Equation to Constraint Set

Our version of **QSIM** automatically generates the set of constraint equations

```
{deriv(v,f1), mult(m,f1,f2), mult(m,g,f3), mult(v,v,f4),
                mult(b,f4,f5), add(f2,f5,f3)}
```

and prompts for initial landmark sets and qualitative states for `f1` ($=dv/dt$) and `v`. We choose initial landmark sets as `{-inf,0,+inf}` for both parameters and set `v` to be `<0,inc>` and `dv/dt` to be `<[0,+inf],dec>`.

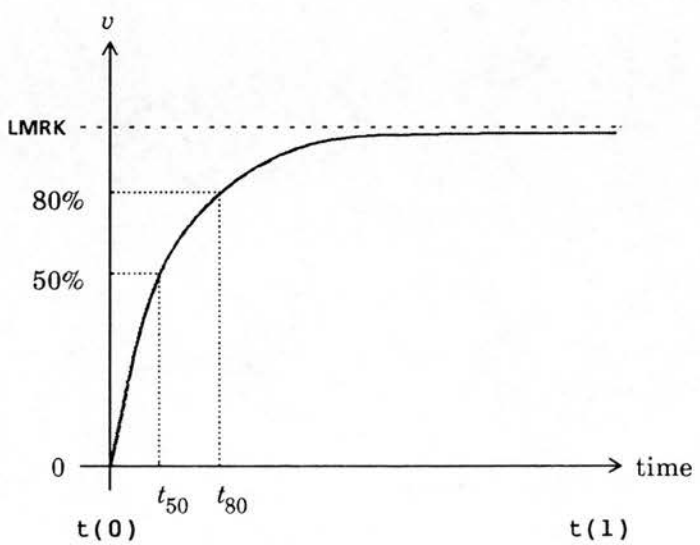


Figure 3-4. Qualitative Behaviour of Falling Stone

Envision Qualitative Behaviour

After revision, this results in the following behaviour prediction:

time	$v(t;b)$	$dv(t;b)/dt$
t(0)	<0,inc>	<[0,+inf],dec>
t(0,1)	<[0,LMRK],inc>	<[0,+inf],dec>
t(1)	<LMRK,std>	<0,std>

Match Qualitative Behaviour to Function Descriptors

From the library, the rising exponential is *congruent* to this behaviour:

name: rising-exponential

conjecture: v = LMRK-(LMRK-0)*exp(-b*t)

consequences: {LMRK>0,
asymptotic-approach(v,LMRK),
horizontal-shift(0),
vertical-shift(0),
vertical-scale(LMRK),
infinite(t(1))}

N.B. the hyperbolic tangent, for $t > 0$, is also congruent to the qualitative behaviour but the system would only explore this possibility in the event of the first trial function proving to be inadequate.

name: rising-hyperbolic-tangent

conjecture: v = 0+(LMRK-0)*tanh(b*t)

consequences: {LMRK>0,
asymptotic-approach(v,LMRK),
horizontal-shift(0),
vertical-shift(0),
vertical-scale(LMRK),
infinite(t(1))}

The residual equation is

$$m*dv/dt - m*g + b*v^2 = R_e \tag{3.2}$$

Each conjecture is associated with a set of schemata which record the numerical substitutions to replace the conjecture for different collocation fractions. Instantiation of the exponential template `F = A - (A - I)*exp(-1*b*t)` results in

trial function: v = LMRK – (LMRK – 0)*exp($-b*t$)

which is tidied to

trial function: v = LMRK*(1 – exp($-b*t$))

with dv/dt = LMRK*b*exp($-b*t$)

Invoke Collocation Schema

Recall that the collocation schemata are obtained by equating the trial function with some fraction of the range. This implicitly (by function inversion) determines a time at which the trial function has completed a certain fraction of the change.

Collocate at 50% completion: LMRK $(1 - \exp(-bt)) = {}^1/_2$ LMRK $\Rightarrow \exp(-bt) = {}^1/_2$ (3.3)

Collocate at 80% completion: LMRK $(1 - \exp(-bt)) = {}^4/_5$ LMRK $\Rightarrow \exp(-bt) = {}^1/_5$ (3.4)

Set Up Residual Equation

The system retrieves this then sets the residual to zero, substitutes the conjecture into the differential equation, evaluates the derivatives, simplifies and passes this to **BOTHER**, a symbolic equation solver, with the goal to isolate b.

At the collocation points $R_e = 0$. Calling the collocation substitutions (in this case for $\exp(-bt_i)$) c_i, equation (3.2) becomes

$$m \text{ LMRK } b\, c_i - mg + b \text{ LMRK}^2 (1 - c_i)^2 = 0 \tag{3.5}$$

Eliminate Discovered Landmarks

Moreover, at t(1), v = <LMRK, std>, dv/dt = <0, std> which on substitution into equation (3.1) yields

$$\text{LMRK} = \sqrt{(mg/b)} \tag{3.6}$$

Isolate Adjustable Parameter(s)

Hence, on substitution into (3.5) implies

$$b = (2 - c_i)\sqrt{(gb/m)} \tag{3.7}$$

Verification

Finally the system must check the sensitivity of this trial function to the choice of collocation point. To do this it computes the ratio of b calculated at different collocation points. The result is

$$b_{50\%} / b_{80\%} = {}^{5}/_{6} \tag{3.8}$$

which passes the verification test established in § 3.4.4.1. suggesting that this trial function is a reasonable hypothesis.

The conjecture embodies a high level description of the qualitative behaviour and the equation for b suggests a relationship between the significant parameters of the problem which is of practical significance in estimating how variations in those parameters are likely to effect behaviour. Once the differential equation, initial condition and *adequacy* criterion are given the approximate solution is found without user intervention.

3.5.2 A Two Parameter Trial Function

This problem is adapted from [Acton & Squire 85 p169]. However our method of solution is quite different. Whereas Acton & Squire reduce the problem to a single adjustable parameter by approximating a slowly varying function (a logarithm) as a constant and integrating the simplified equation directly, we adhere to the **analytic abduction** algorithm which allows for simultaneous optimisation of multiple parameters.

An interesting feature of this example is that it uses the machinery of **QSIM** to perform a *spatial* rather than *temporal* simulation. This is permissible as we are only dealing with one space dimension. By choosing coordinates appropriately the spatial dimension is never negative so, mathematically, the problem is identical to the usual temporal equations **QSIM** handles. Therefore, the "time points" t(1), t(2) *etc* in the following qualitative simulation are in reality space points and space intervals and we re-name them (by hand) to aid readability. However the underlying **QSIM** is unchanged.

Map Differential Equation to Constraint Set

We consider the equation governing the loss of energy of a particle as it penetrates matter. Initially the particle has energy u_0 but this is dissipated as it burrows through a homogeneous medium to depth x. The equation is

$$du/dx = -\log_e(1+u)/u \tag{3.9}$$

This is passed to **QSIM** which automatically abstracts it to the constraint set

```
{deriv(u, f1), M+(f2, u), minus(f2,f3), mult(u, f1, f3)}
```

and then simulates the corresponding system. Notice, in particular, that the logarithmic term is mapped to a weaker monotonic function constraint.

Envision Qualitative Behaviour

One behaviour has the qualitative properties shown in Figure 3-5.

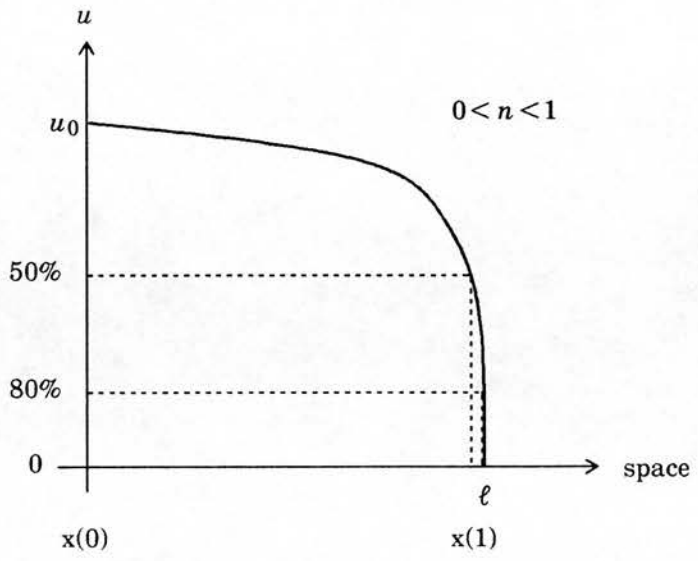


Figure 3-5. Energy Loss as a Particle Penetrates Matter

Qualitatively this is

space	u	du/dx
`x(0)`	`<u0, dec>`	`<[-inf, 0], dec>`
`x(0,1)`	`<[0,u0], dec>`	`<[-inf, 0], dec>`
`x(1)`	`<0,dec>`	`<-inf, dec>`

Match Qualitative Behaviour to Function Descriptors

This matches the template for the binomial function with two adjustable parameters n and ℓ.

trial function: $u = u_0(1 - x/\ell)^n$

with derivative $du/dx = - nu_0/\ell\ (1 - x/\ell)^{n-1}$. For this qualitative form it is known that $0<n<1$ and that $\ell =$ `x(1)`.

Invoke Collocation Schema

Collocate at 50% completion:

$$u_0(1 - x/\ell)^n = u_0/2 \Rightarrow (1 - x/\ell)^n = 1/2 \tag{3.10}$$

Collocate at 80% completion:

$$u_0(1 - x/\ell)^n = u_0/5 \Rightarrow (1 - x/\ell)^n = 1/5 \tag{3.11}$$

Set Up Residual Equation

Calling the collocation substitution c_i, the system sets up the equation residual

$$-(nu_0/\ell)*(1 - x/\ell)^{n-1} + [\log_e(1 + u_0(1 - x/\ell)^n) / (u_0(1 - x/\ell)^n)] = R_e \tag{3.12}$$

which, being zero at the collocation points, becomes

$$(nu_0^2/\ell)\, c_i^{2-(1/n)} = \log_e(1 + u_0 c_i) \tag{3.13}$$

It then considers a pair of collocation substitutions c_1 and c_2. Above for collocating at 50% and 80% completion these were $c_1 = 1/2$ and $c_2 = 1/5$. However, in the implementation these are left as free variables to facilitate examining the consequences of changing their values.

$$(nu_0^2/\ell) = c_1^{(1/n)-2} \log_e(1 + c_1 u_0) \tag{3.14}$$

$$(nu_0^2/\ell) = c_2^{(1/n)-2} \log_e(1 + c_2 u_0) \tag{3.15}$$

The next stage is to set up the goal to isolate the adjustable parameters, n and ℓ.

Isolate Adjustable Parameter(s)

Solving for n

Divide (3.14) by (3.15)

$$(c_2/c_1)^{(1/n)-2} = \log_e(1+c_1 u_0) / \log_e(1+c_2 u_0)$$

Taking logarithms of either side and inverting

$$n = \log_e(c_2/c_1) / [2 \log_e(c_2/c_1) + \log_e(\log_e(1+c_1 u_0) / \log_e(1+c_2 u_0))\ ] \qquad (3.16)$$

(†)

All terms are numerically evaluable except the double logarithm (†). **BOTHER** estimates a value for such terms by computing upper and lower limits if possible. This is fully explained in *Chapter 8*. For now we merely cite the result.

If u_0 is small $\log_e([\log_e(1+c_1 u_0) / \log_e(1+c_2 u_0)]) \simeq \log_e(c_1/c_2)$

If u_0 is large $\log_e([\log_e(1+c_1 u_0) / \log_e(1+c_2 u_0)]) \simeq \log_e(1) \simeq 0$

Therefore the range of n (assuming it behaves monotonically between large and small u_0) is

$$1/2 < n < 1$$

Acton & Squire tell us that u_0 is indeed large so $n \simeq 1/2$. However, this knowledge can neither be deduced from the equation nor assumed to be always available in a problem statement. The important point is that if this information were known to **BOTHER** it could use it to deduce the correct approximation. But if such information is not available **BOTHER** works a problem hard to try to impose the tightest bounds possible.

BOTHER always inspects the relative difference between two bounds. If this is less than 33%, **BOTHER** proposes their average as an acceptable compromise value. In the present case the bounds are too dissimilar to sanction this operation. Instead, **BOTHER** returns the lower of

upper bounds as shown and two chains of inference are spawned, one taking $n = 1/2$ and the other taking $n = 1$.

Solve for ℓ

The next step is to solve for ℓ, the other adjustable parameter in the trial function. This is done by isolating ℓ in equation (3.13) resulting in

$$\ell = c_i^{2-(1/n)} n u_0^2 / [\log_e(1 + u_0 c_i)] \tag{3.18}$$

which unfortunately depends on the collocation substitution c_i. *N.B.* Acton & Squire's solution to this problem is obtained by other means and yields an approximate expression for ℓ which also depends on the collocation points. A compromise is to take a value for c_i in between c_1 and c_2.

We saw above that both bounds on n were retained explicitly. This defines two possible contexts for computing an approximate expression for ℓ. Taking $n = 1/2$ (*i.e.* u_0 large implicitly) it is found that

$$\ell = u_0^2 / [2\log_e(1 + u_0 c_i)] \tag{3.19}$$

Taking $n = 1$ (*i.e.* u_0 small implicitly)

$$\ell = c_i u_0^2 / [\log_e(1 + u_0 c_i)] \tag{3.20}$$

Verification

Next the system determines how sensitive the values of n and ℓ are to the particular choice of collocation fractions c_1 and c_2. n is clearly not affected because all dependancies of c_1 and c_2 cancel. Thus $1/2 < n < 1$ is an *adequate* estimate.

For ℓ it is necessary to consider the two possibilities of u_0 being large or small in addition to different choices for c_1 and c_2. Let ℓ_1 and ℓ_2 be the values of ℓ at c_1 and c_2 respectively. Taking u_0 large and substituting in equation (3.19), the system derives

$$\ell_1 / \ell_2 = \log_e(1+u_0c_2) / \log_e(1+u_0c_1) \simeq [\log_e(u_0) + \log_e(c_2)] / [\log_e(u_0) + \log_e(c_1)] \simeq 1$$

Similarly, taking u_0 small, equation (3.20) implies

$$\ell_1 / \ell_2 = [c_1 \log_e(1+u_0c_2)] / [c_2 \log_e(1+u_0c_1)] \simeq (c_1/c_2)*(c_2/c_1) = 1$$

Hence the trial function is a good approximation for both large and small u_0, but expression for ℓ is different in each case. In the particular example, where u_0 corresponds to the initial kinetic energy of a subatomic particle, it is physically sensible to suppose u_0 is indeed large. In this case it appears that the precise value of ℓ is insensitive to the exact choice of collocation fraction. Hence we accept the trial function

trial function: $u = u_0(1 - x/\ell)^n$

with n and ℓ as above as a reasonable approximate solution to equation (3.9). This provides an approximate functional relationship between energy and penetration depth which is essential for the design of detection equipment and gives an explicit assumption under which the approximation is valid (*i.e.* $u_0 \gg 1$)..

3.5.3 Harmonic Balance

As a third example, consider the motion of a ball sliding in a U-tube shaped to produce a restoring force proportional to the cube of the displacement. The initial state description is $u = u_0$ when $t = t(0)$ and equation of motion is

$$d^2u/dt^2 + cu^3 = 0 \qquad (3.21)$$

Again, this example is taken from a set of unsolved end of chapter problems in [Acton & Squire 85 p90].

Map Differential Equation to Constraint Set

This is automatically mapped into the six **QSIM** constraints

```
{deriv(u,f1), deriv(f1,f2), mult(u,u,f3), mult(u,f3,f4),
                mult(c,f4,f5), add(f2,f5,0)}
```

Envision Qualitative Behaviour

A behaviour predicted from these constraints is as shown in Figure 3-6.

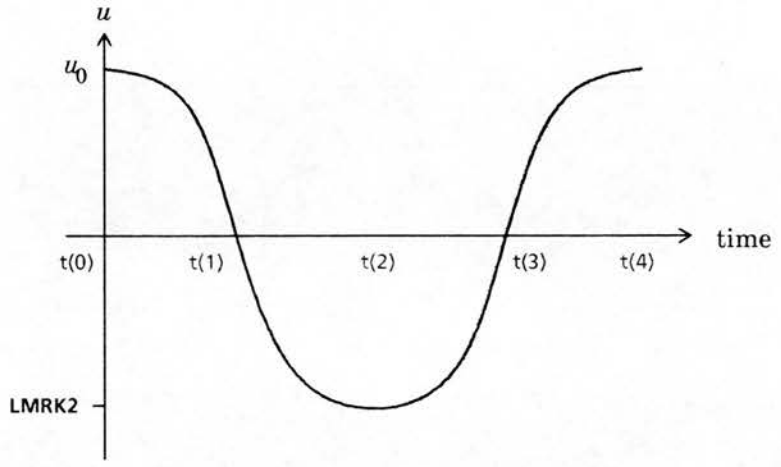


Figure 3-6. Qualitative Behaviour of a Ball in a U-tube

After revision, this is represented as the following sequence of state transitions:

time	$u(t;b)$	$du(t;b)/dt$
t(0)	<u0,std>	<0,dec>
t(0,1)	<[0,u0],dec>	<[LMRK1,0],dec>
t(1)	<0,dec>	<LMRK1,std>
t(1,2)	<[LMRK2,0],dec>	<[LMRK1,0],inc>
t(2)	<LMRK2,std>	<0,inc>
t(2,3)	<[LMRK2,0],inc>	<0,LMRK3],inc>
t(3)	<0,inc>	<LMRK3,std>
t(3,4)	<[0,u0],inc>	<[0,LMRK3],dec>
t(4)	<u0,std>	<0,dec>

Match Qualitative Behaviour to Function Descriptors

The above behaviour matches the function descriptor of a cosinusoidal trial function,

trial function: $u = u_0\cos(bt)$

with derivatives $du/dt = -bu_0\sin(bt)$ and $d^2u/dt^2 = -b^2u_0\cos(bt)$.

Invoke Harmonic Balance Schema

As the behaviour is oscillatory harmonic balance is activated.

Set Up Residual Equation

On substitution into the differential equation, rewriting $\cos^3(bt)$ as $^1/_4(3\cos(bt)+\cos(3bt))$ (**BOTHER** knows of such expansions) and arranging into ascending order of multiple angles, the equation residual becomes:

$$(-b^2u_0+{}^3/_4u_0{}^3c)\cos(bt) + {}^1/_4\,cu_0{}^3\cos(3bt) = R_e \qquad (3.22)$$

Isolate Adjustable Parameter(s)

As there is only one adjustable parameter in the conjecture the system can only balance the $\cos(bt)$ term. Hence $(-b^2u_0+{}^3/_4u_0{}^3c)$ is set to zero and the goal is to isolate b. After simplification this results in

$$b = u_0\,((\sqrt{3})/2)\,\sqrt{c} \qquad (3.23)$$

Eliminate Discovered Landmarks

The newly discovered landmark, **LMRK2**, is readily eliminated by inspecting the "consequences" attribute of the cosinusoidal trial function object.

consequences: $\{u_0 = -$ **LMRK2**, horizontal-shift(0), vertical-shift(0), vertical-scale(u_0), period(t(4)), parameters([b])$\}$.

Verification

For Harmonic Balance, verification consists of examining the ratio of the amplitude of the lowest neglected subterm to that of the restoring term in (3.21). That is,

$$\max(\,|{}^1/_4\, cu_0{}^3\cos(3bt)|\,) \,/\, \max(\,|cu_0{}^3\cos(bt)|\,) = {}^1/_4$$

The adequacy criterion was that this ratio should be less than $^1/_3$. As it is, the trial function is deemed *adequate*.

Again the key point is that from a differential equation and a purely qualitative simulation it has been possible to compose a high level description of the behaviour ($u = u_0\cos(bt)$) which highlights relationships between the problem parameters.

3.5.4 First Guesses Can Fail

So far we have been lucky: our first trial functions have been adequate. However, this will not always be the case as the example below illustrates.

The equation governing the chemical reaction A + 2B → C is

$$dn_c/dt = K*(n_a - n_c)*(n_b - 2*n_c)^2 \tag{3.24}$$

where n_a = initial number of molecules of A, n_b = initial number of molecules of B and n_c = the number of molecules of C formed by time t.

Map Differential Equation to Constraint Set

This equation is abstracted to the **QSIM** constraint set

{deriv(n_c,f1), mult(2,n_c,f2), add(f2,f3,n_b), mult(f3,f3,f4),
add(f5,n_c,n_a), mult(f5,f4,f6), mult(K,f6,f1)}

Envision Qualitative Behaviour

Qualitative simulation of this system of equations reveals that initially the rate of formation of C is high but gradually falls off as the number of free molecules of reactants declines.

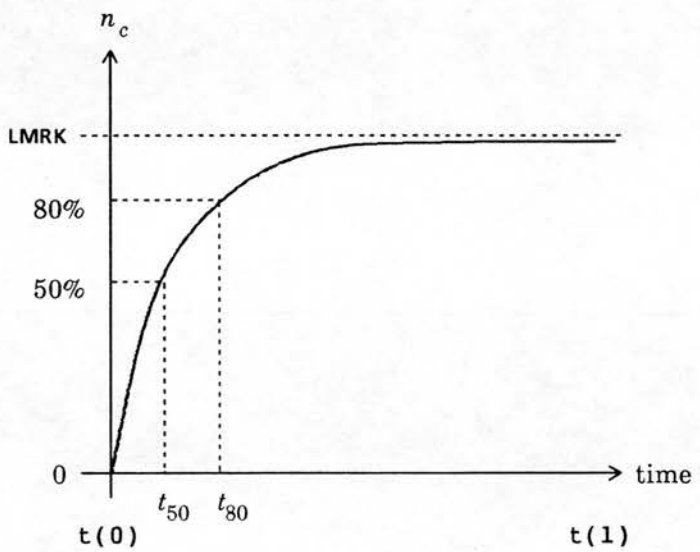


Figure 3-7. Qualitative Behaviour of the Chemical Reaction A + 2B→ C

The qualitative behavioural description is thus

time	n_c(t;b)	dn_c(t;b)/dt
t(0)	<0,inc>	<[0,+inf],dec>
t(0,1)	<[0,LMRK],inc>	<[0,+inf],dec>
t(1)	<LMRK,std>	<0,std>

Match Qualitative Behaviour to Function Descriptors

which is isomorphic to the previous prediction for the falling stone example and consequently activates the same initial trial function hypothesis (suitably instantiated and tidied)

trial function: $n_c = \text{LMRK}\,(1 - \exp(-bt))$

Invoke Collocation Schema

Following the collocation procedure exactly as before we collocation takes place at 50% and 80% completion. The corresponding substitutions, c_i, are

	$\exp(-bt)$
Collocate at 50%	$1/2$
Collocate at 80%	$1/5$

Eliminate Discovered Landmarks

As in §3.5.1 it is deduced that the landmark value **LMRK** $= n_b/2$.

Set Up Residual Equation

On substitution into the residual equation

Isolate Adjustable Parameter(s)

the system isolates the adjustable parameter b_i (*i.e.* b at a particular collocation fraction) to yield

$$b_i = 2Kn_an_bc_i(1-(n_b/2n_a)(1-c_i)).$$

Verification

The next step is to evaluate the b_i and their ratio. The details of this procedure would intrude too much to incorporate them here. Instead we simply note that the necessary simplifications require both meta-level concepts such as those found in **PRESS** and order of magnitude reasoning over complicated terms and functions. We will return to this example in *Chapter 8* to show how **BOTHER** performs these inferences. Here we simply cite the results so obtained. For the exponential

$$b_{50\%} = Kn_an_b(1-1/4\,(n_b/n_a))$$

$$b_{80\%} = (2/5)\,Kn_an_b(1-2/5\,(n_b/n_a))$$

To verify the trial function we compute the ratio

$$b_{50\%}/b_{80\%} = (5/2)\,(1 + (3/20)n_b/n_a)$$

and deduce that as it is in excess of $5/2$ the original trial function hypothesis is *poor*.

The next section suggests what to do when confronted by such an impasse.

3.6 Recovering From Failure

3.6.1 Iteration in Function Space

If, during the verification stage, the conjecture is found to be an inadequate approximation to the true solution the system needs to be able to improve upon it. Surprisingly, Acton & Squire avoid this issue altogether and offer no advice. The traditional approach would be to add in another trial function from the basis set and repeat the weighted residual procedure using the extended trial function.

We do not adopt this technique. **Analytic abduction** was inspired by a wish to discover approximate solutions to differential equations which were readily comprehensible. A trial function containing a sum of basis functions each with its own adjustable parameters hardly meets this criterion. Instead we use the predicted qualitative behaviour of the solution and various abstractions of it to guide the system through the space of possible function conjectures. We have devised three methods for doing this:

- *topological isomorphism*
- *topological similarity* and
- *exaggeration*

The first performs no abstraction of the behaviour. The second abstracts over point convexities and the third abstracts over the qualitative value of the first derivative at the initial time point.

3.6.2 Topological Isomorphism

In §3.5.4 the trial function, n_c = LMRK (1 – exp(– bt)) failed the verification stage. The first attempt at recovering from such a failure involves finding another trial function which is also *congruent* to the qualitative behavioural description and which proves to be a better approximation.

Definition 3-1: *congruence*

Two trial functions are *congruent* if and only if they have the same sequence of critical points, convexities and bounding intervals.

In order to test whether two trial functions are *congruent* a necessary and sufficient condition is that their function descriptors unify.

Having found a *congruent* trial function, the **analytic abduction** procedure is followed exactly as before.

3.6.3 Topological Similarity

Topological isomorphism demands that the function descriptor of the second trial function is identical to that of the first. However, it is possible to relax strict equality a little without losing too much information by abstracting over "point-convexities" in the following sense: Given two trial functions, let the qualitative values of their convexities match at their time points according to the rules

$$\texttt{inc} \quad \stackrel{?}{=} \quad \texttt{inc} \vee \texttt{std}$$

$$\texttt{std} \quad \stackrel{?}{=} \quad \texttt{inc} \vee \texttt{std} \vee \texttt{dec}$$

$$\texttt{dec} \quad \stackrel{?}{=} \quad \texttt{dec} \vee \texttt{std}$$

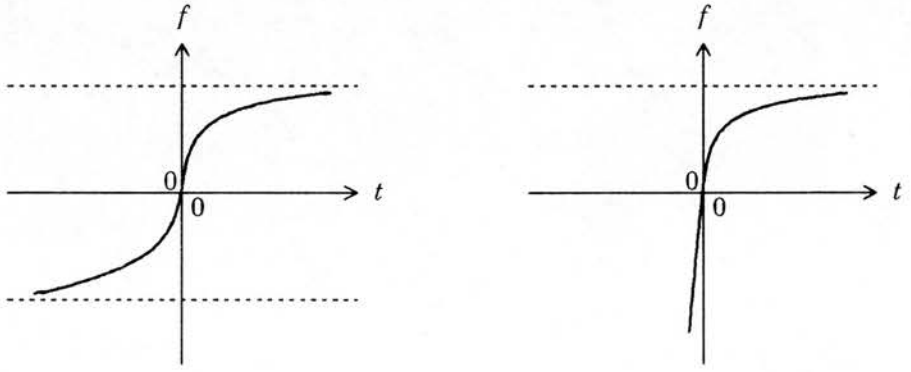
(where "$\stackrel{?}{=}$" means "matches successfully"). In other words at time points let **std** match any qualitative value. Over time intervals, however, the convexities must be syntactically identical to match. The rationale being that a point difference in convexity is tolerable but an interval difference is not. This is an implicit inference Acton & Squire use. It needs to be made explicit to facilitate the computer implementation of **analytic abduction**. This abstraction over point convexities allows a alternative definition of congruence.

Definition 3-2: *almost everywhere congruence*

Two trial functions are *almost everywhere congruent* if and only if they have the same sequence of critical points, the same convexities over time intervals, compatible convexities at time points and the same bounds.

Hence trial functions which are *almost everywhere congruent* but not *congruent* have a different qualitative value for the convexity at some point. The concept of "*almost everywhere congruence*" is therefore weaker than that of "*congruence*".

For example, the functions $1-\exp(-bt)$ and $\tanh(b't)$ are *almost everywhere congruent* for $t \geq 0$ but *congruent* for $t > 0$.



$f = \tanh(b't)$ $\qquad$ $f = (1\text{-}\exp(\text{-}bt))$

Figure 3-8. Example of Almost Congruent Trial Functions for t ≥0

Almost everywhere congruence allows a notion of *topological similarity* to be defined which preserves the qualitative values for the function and its first derivative over all time points and intervals. After presenting an example where *topological similarity* is used to find an improved guess for the behaviour of the chemical reaction discussed in §3.5.4 we will discuss an alternative abstraction of the qualitative behaviour which modifies the qualitative values of the first derivative at the initial time point.

3.6.3.1 Topological Similarity Example

For the example of §3.5.4, as $t>0$, a second trial function would be

trial function: $n_c = \textsf{LMRK} \tanh(b't)$

again with $\textsf{LMRK} = n_b/2$. Following the collocation procedure exactly as before, collocation is performed at 50% and 80% completion. The corresponding substitutions, c_i, are

	$\tanh(b't)$
Collocate at 50%	$^1/_2$
Collocate at 80%	$^4/_5$

and

$$b'_i = 2Kn_an_b(1 - c_i\,(n_b/2n_a))\,(1 - c_i)/(1+c_i).$$

Hence, the value of the adjustable parameter computed at different collocation fractions is

$$b'_{80\%} = (^1/_3)\,Kn_an_b(1 - (^2/_5)(n_b/n_a))$$

$$b'_{50\%} = (^2/_3)\,Kn_an_b(1 - (^1/_4)(n_b/n_a))$$

and their ratio is

$$b'_{50\%}/b'_{80\%} = 2(1 + {}^3/_{20}(n_b/n_a))$$

which, although a better approximation, is still not *adequate* with respect to the original differential equation.

Hence, merely changing the trial function will not always be sufficient to guarantee success.

3.6.4 Exaggerated Initial Condition

The third technique is invoked if an *adequate* conjecture still cannot be found. If the *topological isomorphism* and *topological similarity* methods fail, there can be no function in the library of standard forms which is both *congruent* or *almost everywhere congruent* with respect to the qualitative behaviour and *adequate* with respect to the differential equation. Something clearly has to give. The idea behind the current technique is to exaggerate the initial condition and look for an approximation *congruent* to the new qualitative description but *adequate* with respect to the original equation. Intuitively, the precision at the initial condition is relaxed in return for finding a better fit over most of the curve.

Definition 3-3. *exaggeration*

One trial function, T_1, is an *exaggeration* of another, T_2, if and only if T_1 is *congruent* to T_2 apart from having a positive (negative) infinite gradient, instead of a positive (negative) finite gradient, at the initial time point.

Exaggeration is not an *ad hoc* procedure: the given initial condition is not replaced with anything we please. Exaggeration preserves function values and convexities but scales initial derivative values. For example, if the initial condition is

time	function	derivative
t=t0	<0,inc>	<[0,+inf],dec>

the exaggerated initial condition is

time	function	derivative
t=t0	<0,inc>	<+inf,dec>

In other words positive declining gradients are mapped to infinite declining ones. Similar results apply for negative gradients. To find the approximate solution to the original differential equation we simply recurse on the **analytic abduction** procedure using the exaggerated qualitative description.

Of course it is important to remember that the new conjecture will be a poor approximation to the exact solution at very early times. However, in most applications the parameters of interest will be those defining a time scale for a process (*e.g.* a rise time for a voltage edge or a frequency). Exaggeration allows us to find an approximate function which adequately describes most of the behaviour. Therefore the parameters determining time scales are valid approximations of the real behaviour.

3.6.4.1 Exaggeration Example

We saw before (example 3.5.4) how verification rejected n_c = LMRK (1 – exp(– at)) as an approximation of the solution of

$$dn_c/dt = K*(n_a - n_c)*(n_b - 2*n_c)^2$$

Exaggeration allows us to find a better approximation which passes the verification tests. At the point *exaggeration* is called upon, the system will have already deduced the qualitative properties of the solution.

Exaggerate Initial Condition

Hence the first thing it must do is to exaggerate the initial condition.

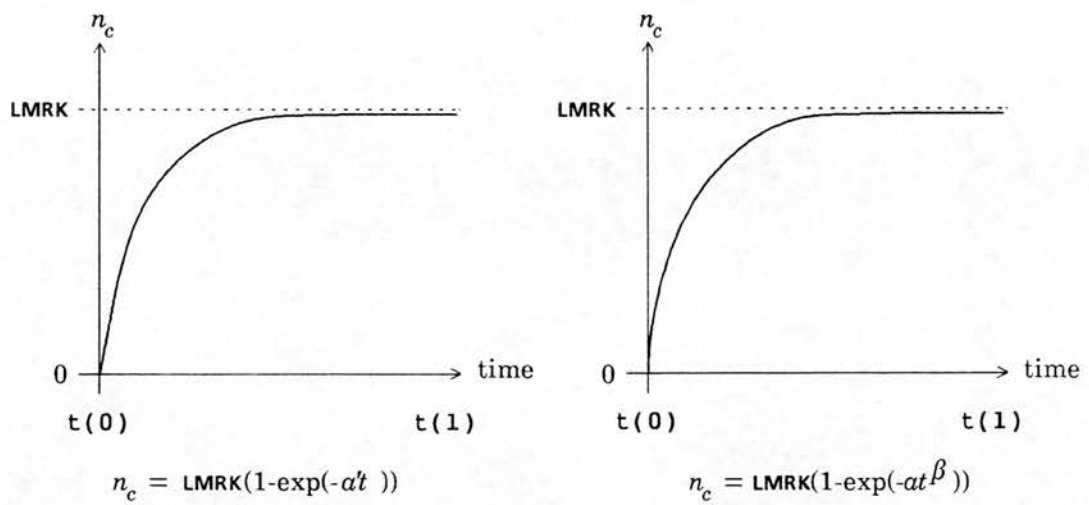
	time	function	derivative
normal	t=t0	<0,inc>	<[0,+inf],dec>
exaggerated	t=t0	<0,inc>	<+inf,dec>

Match Qualitative Behaviour to Function Descriptors

The new qualitative description permits matching to functions whose initial gradients are infinite but thereafter become finite and resemble the exponential rise curves over most of their range. In particular it is possible to match the exaggerated qualitative behaviour to

trial function: $n_c = \text{LMRK}\,(1 - \exp(-at^{\beta}))$

Qualitatively, the graphs of LMRK(1–exp(–$a't$)) and LMRK(1–exp(–at^{β})) are very similar apart from the initial gradient difference.



$n_c = \text{LMRK}(1\text{-exp}(\text{-}a't))$ $\qquad$ $n_c = \text{LMRK}(1\text{-exp}(\text{-}at^{\beta}))$

Figure 3-9. Effects of Exaggeration

This trial function contains two adjustable parameters, a and β, requiring two collocations. This will necessitate more computation but permits finer "tuning" to fit the differential equation.

The derivative of this trial function is

$$dn_c/dt = \text{LMRK}a\beta t^{\beta-1}\exp(-at^{\beta})$$

which, for $\beta<1$, is infinite at $t=0$.

The task is to find values for α and β such that LMRK$(1-\exp(-\alpha t^{\beta}))$ is a better approximation to (3.24) over most of the range than LMRK$(1-\exp(-\alpha' t))$ was.

The system follows the standard collocation algorithm.

Invoke Collocation Schema

There are two adjustable parameters in the trial function so two collocations are required. The decision to collocate at 33% and 80% completion was made under the *coverage* heuristic. This means the collocation fractions are $f_1 = 1/3$ and $f_2 = 4/5$. As before, to facilitate later verification, the system defers substitution of values for the f_i.

$$\text{Collocate at } f_i*100\%: \quad \text{LMRK} * (1-\exp(-\alpha t_i^{\beta})) = f_i * \text{LMRK}$$

which, letting $c_i = 1 - f_i$, implies

$$\exp(-\alpha t_i^{\beta}) = c_i \quad \rightarrow \quad t_i = [1/\alpha \log_e(1/c_i)]^{1/\beta} \tag{3.25}$$

Had the trial function happened to be the exact solution, α and β would be the same for all collocation fractions. However, as the conjecture is only approximate this will not be so. The trick is to find a conjecture for which the variation in α and β is within the criterion specified in §3.4.4 (usually 20% or so).

Eliminate Discovered Landmarks

At `t(1)`, n_c = `<LMRK,std>`, dn_c/dt = `<0,std>` which implies, from equation (3.24), LMRK $= n_b/2$ (as $n_b < 2n_a$).

Set Up Residual Equation(s)

As there are two adjustable parameters the system sets up residual equations for two collocation fractions with the goal to solve simultaneously for α and β. The symbolic manipulations involved are fairly straightforward. The most interesting feature is a "back-of-the-envelope" reasoner (**BOTHER**) which we discuss in *Chapter 8*. Back of the

envelope reasoning allows considerable simplification of expressions by relying on powerful, but controlled, order of magnitude inferences.

Substituting (3.25) into (3.24) for collocation fractions c_1 and c_2 and using **LMRK** $= n_b/2$ yields the simultaneous equations

$$a^{1/\beta}\beta\,(\log_e(1/c_1))^{1-1/\beta} = 2Kn_an_bc_1(1-(n_b/2n_a)(1-c_1)) \tag{3.26}$$

$$a^{1/\beta}\beta\,(\log_e(1/c_2))^{1-1/\beta} = 2Kn_an_bc_2(1-(n_b/2n_a)(1-c_2)) \tag{3.27}$$

Isolate Adjustable Parameter(s)

Solve for β

To solve for β the system divides (3.26) by (3.27) and takes the logarithm of either side to give:

$$(1-1/\beta)\log_e(\log_e(c_1)/\log_e(c_2)) = \log_e(c_1/c_2) + \log_e(\,(1-(n_b/2n_a)(1-c_1))/(1-(n_b/2n_a)(1-c_2))\,)$$

This expression is passed to **BOTHER** to see if can be simplified. The second term on the right is the troublesome one as all the others are purely numeric or contain β in an isolatable position.

As we describe **BOTHER** and the particular details of this example in *Chapter 8* we will omit them here. The key point is that **BOTHER** imposes the weakest assumptions sufficient to compute a numerical estimate for terms containing symbolic constants of unknown absolute value. This requires a combination of term rewriting and inequality reasoning.

In the present example, **BOTHER** rewrites the expression for β as

$$\beta = [\log_e(\log_e(c_1)/\log_e(c_2))] / [\log_e(\log_e(c_1)/\log_e(c_2)) - \log_e(c_1/c_2) - \log_e(1+(c_1-c_2)*(n_b/2n_a))]$$

with $c_1 = 2/3$ and $c_2 = 1/5$ and bounds the term containing $n_b/2n_a$ as

$$0 < \log_e(1+(c_1-c_2)*(n_b/2n_a)) < c_1-c_2$$

Hence, the entire expression for β can be bounded by

$$0.452 < \beta < 0.534$$

BOTHER therefore suggests an approximate value of $\beta = 0.49$.

Solve for a

To calculate a the system substitutes $\beta = 0.49$ into equation (3.26) with the goal to isolate a. It chose equation (3.27) (*i.e.* the equation in $c_2 = 1/5$) rather than equation (3.26) because we have a heuristic which tells the system to avoid using parameter values derived from collocations nearest to the initial state of exaggerated behaviours. The justification for this heuristic is that we are aware that the trial function will be a better approximation further from the start of the curve. This gives

$$a^{2.04}(0.49)\,(\log_e(5))^{-1.04} = 2Kn_an_b(1/5)(1-(n_b/2n_a)(1-1/5))$$

which simplifies to

$$a = 1.15\,[\,Kn_an_b(1-2n_b/5n_a)\,]^{1/2.04}$$

Verification

The next phase is to test whether the new trial function is a better approximation than the old one by varying the collocation point. Collocating at half completion, the collocation fraction is f_3*LMRK $= 1/2$*LMRK which implies writing $f_3 = 1-c_3$,

$$\exp(-at_3^{\beta}) = c_3 \quad \rightarrow \quad t_3 = [1/a\,\log_e(1/c_3)]^{1/\beta} \tag{3.28}$$

and hence,

$$a^{1/\beta}\beta\,(\log_e(1/c_3))^{1-1/\beta} = 2Kn_an_bc_3(1-(n_b/2n_a)(1-c_3)) \tag{3.29}$$

This allows new values to be computed for a and β and to test that they are not too different from the values computed using other collocation points.

New β

β is fairly insensitive to the choice of collocation point as it is built from logarithmic or doubly logarithmic terms. To verify this the system computes the two new values for β one from the pairing of equations (3.26) and (3.29) and the other from the pairing of equations (3.27) and (3.29). As the calculation is entirely analogous to that above we simply state the result.

Equations	c_i	c_j	**Range of β**	**Average β**
(3.26)&(3.27)	$2/3$	$1/5$	$0.452 < \beta < 0.534$	0.49
(3.26)&(3.29)	$1/2$	$1/5$	$0.409 < \beta < 0.479$	0.44
(3.27)&(3.29)	$2/3$	$1/2$	$0.541 < \beta < 0.651$	0.60

The last entry is calculated from collocating at one third and one half completion. This is likely to be the poorest approximation as the initial gradient has been exaggerated. Hence a larger error here is to be expected. Bearing this in mind, the variation of β is within acceptable bounds.

New a

Similarly, it is necessary to re-compute values for a using each pair of collocation points and their associated β values. Solving simultaneously for a yields

$$a = (c_i c_j)^{\beta/2}/[\beta^{\beta}\,(\log_e(c_i)\log_e(c_j))^{(\beta-1)/2}](2Kn_a n_b)^{\beta}[(1-(n_b/2n_a)(1-c_i))\,(1-(n_b/2n_a)(1-c_j))]^{\beta/2}$$

This is obtained by considering the product of any pair of equations (3.26), (3.27) or (3.29) and isolating a. We set **BOTHER** to work on this equation to see how it can be simplified. Since we know for this trial function that $\beta < 1$, $n_b < 2n_a$, $0 < 1 - c_i < 1$, $0 < 1 - c_j < 1$ and $c_i > c_j$ **BOTHER** is able to approximate the product on the right to yield

$$a = (c_i c_j)^{\beta/2}/[\beta^{\beta}\,(\log_e(c_i)\log_e(c_j))^{(\beta-1)/2}]\,(2Kn_a n_b)^{\beta}[1-\beta(n_b/2n_a)(1-0.5*(c_i+c_j))] \quad (3.30)$$

Using (3.30) the following formulae are obtained for a computed from different pairs of collocation fractions. If the solution were exact, a would be the same.

Equations	c_i	c_j	**Average** β	a
(3.26)&(3.27)	$^2/_3$	$^1/_5$	0.49	$0.78\,(2Kn_an_b)^{0.49}[1-0.204\,(n_b/2n_a)]$
(3.26)&(3.29)	$^1/_2$	$^1/_5$	0.44	$0.89\,(2Kn_an_b)^{0.44}\,[1-0.286\,(n_b/2n_a)]$
(3.27)&(3.29)	$^2/_3$	$^1/_2$	0.60	$0.76\,(2Kn_an_b)^{0.60}[1-0.250\,(n_b/2n_a)]$

These formulae look encouraging in the sense that respective parameters are comparable in magnitude but they cannot easily be used to ascertain whether the variation in a is less than it was for the earlier trial function. This is because the quotient of any two a's contains an inestimable constant $(2Kn_an_b)^m$ with $0.44 \leq m \leq 0.60$.

The problem of having to estimate this term would have been eliminated had a single value for β been used rather than the values computed for the appropriate pair of collocation points. In this case the index m would be the same in all the formulae for a and hence the inestimable term would cancel out from the quotient of any two a's. We call a set of formulae *inspectable* if the quotient of any pair reduces, *via* **BOTHER**, to a number.

Taking β to be the average of its values at different collocation points it is found that

$$\beta = 0.51$$

Using this value in equation (3.30) above yields a revised set on values for a which are *inspectable*.

Equations	c_i	c_j	**Average** β	a
(3.26)&(3.27)	$^2/_3$	$^1/_5$	0.51	$0.76\,(2Kn_an_b)^{0.51}[1-0.289\,(n_b/2n_a)]$
(3.26)&(3.29)	$^1/_2$	$^1/_5$	0.51	$0.80\,(2Kn_an_b)^{0.51}\,[1-0.332\,(n_b/2n_a)]$
(3.27)&(3.29)	$^2/_3$	$^1/_2$	0.51	$0.78\,(2Kn_an_b)^{0.51}[1-0.213\,(n_b/2n_a)]$

By considering the ratio of any pair of these formulae for a calculated at different collocation points it is possible to verify that the predicted value for a is within acceptable error limits.

Thus we conclude that the a reasonable approximation over most of the range to the solution of

$$dn_c/dt = K(n_a - n_c)(n_b - 2*n_c)^2$$

is

$$n_c = \tfrac{1}{2}n_b(1-\exp(-[0.78\,(2Kn_an_b)^{0.51}(1 - 0.28\,(n_b/2n_a))]\,t^{0.51}))$$

which **BOTHER** may simplify even further to

$$n_c = \tfrac{1}{2}n_b(1-\exp(-[0.78\,(2Kn_an_b)^{0.51}]\,t^{0.51}))$$

as $0.28*n_b/2n_a \ll 1$.

We re-iterate that this is a good fit for the *exaggerated* initial condition and cannot be expected to be accurate at early times. However, because it is a good fit *overall*, the parameters which determine a timescale yield reasonable estimates. Moreover, notice that the solution is readily visualised and suggests a functional relationship between significant system parameters.

This trade-off between precision and definiteness of statement is very common in engineering. This kind of reasoning can be extremely subtle being sensitive to the order in which approximations are made in addition to the legitimacy of those approximations. We return to this issue in *Chapter 8*.

3.7 The Choice of Standard Functions

A central issue in the **analytic abduction** procedure is how to choose which functions to store in the library of "standard forms".

Recall that any function can be expanded as the sum of orthogonal functions. Traditional weighted residual methods tackle the problem of which functions to regard as "standard" by having library entries which are progressively higher members from a set of orthogonal functions. Then the exact solution is approximated by a finite sum of such functions.

For our application we seek one term approximations so the question of which functions to regard as standard is paramount.

Mathematically speaking there are an infinite number of possible qualitative forms. Consequently, if we adopted a naïve approach of a one to one correspondence between behaviours and functions, would require an infinite number of library entries.

Fortunately, many equations arising from real world situations have reasonably well behaved solutions due to the action of domain invariants such as conservation of energy. Such invariants tend to enforce continuity and regularity on behavioural trends *e.g.* the envelope of an oscillation is usually either monotonic or a simpler oscillation rather than a sequence of random maxima and minima. There are exceptions, of course, such as chaotic systems, which are more appropriately handled using phase space approaches. These are an important class of systems but our inability to deal with them does not nullify our approach because there are still many systems whose behaviour is sufficiently regular. Indeed mathematics so far has not yielded any tools for doing anything useful with the recognition that a system is behaving chaotically in the sense that the observation alone does not assist in predicting the system's subsequent behaviour.

This still leaves an enormous number of regular functions to consider and possibly encode in the library. Acton & Squire propose three basic qualitative forms which they claim describe the vast majority of the qualitative behaviours of real systems [Acton & Squire 85 p6]. The *exponential form* captures the notion of transition between two states; the *parabolic form* captures the notion of symmetry about an extremum and the *sinusoidal form* captures the

notion of stable oscillation. This is by far the weakest part of their procedure and is ill-principled.

Our approach aimed to develop a more expressive and principled set of trial functions. We selected functions with various principles in mind.

Principle 1: The trial functions should be sufficiently simple so that their qualitative behaviour may readily be comprehended and the functions easily manipulated.

Principle 2: For each qualitatively distinct behaviour there should be at least two linearly independent analytic functions *congruent* to it.

Principle 3: Regular behaviours are usually monotonic, have a small number of critical points and approach an asymptote or oscillate infinitely. The library should contain special functions possessing these features.

Principle 4: If an unfamiliar qualitative behaviour is encountered we should not fail ignominiously but should attempt to retrieve as much partial information as possible. This idea led to **segment calculus,** which we discuss in the next chapter.

Principle 5: We should pay due regard to the "classics" of mathematical physics. Some functions recur so frequently it is often possible to bias one's guesses towards these functions. Examples include the exponential, trigonometric functions, gamma function, binomial distributions, Bessel functions, logistic growth curves *etc.*

Principle 6: If we want to adhere to cognitive plausibility we should consider what is reasonable for a typical engineer or physicist to know.

Principle 7: The adjustable parameters should be inserted into the trial function so as to allow a wide range of behaviour simply by varying their values.

Principle 8: The library is *complete* modulo n critical points *i.e.* for some choice of positive integer n, there exists a standard form for any function containing m critical points ($0 \leq m \leq n$). Each extension of the library therefore requires new entries for all qualitatively distinct functions containing $n+1$ critical points.

3.7.1 Library

The size of the library is governed by the maximum number of critical points tolerated in the qualitative behaviour. As the maximum number of critical points tolerated increases, the library will grow exponentially. In the next chapter, we consider what to do if we generate a behaviour containing more critical points than are to be found in any library entry.

3.7.2 Completeness Modulo n

We want to ensure that for some number, n, that we generate all qualitatively distinct behaviours of functions containing n critical points or less. To do this we define a grammar of curve segments which dictates how different shapes of curve segments may join together. Given n, the grammar can be used to enumerate, in principle, all qualitatively distinct possibilities. However pragmatic considerations make it hard to even guarantee completeness modulo 1. Behaviours containing more critical points need special treatment.

To see how the number of possibilities grow with increasing n we list the number of qualitatively distinct behaviours that are accessible from an initial segment shown below (a complete catalogue of distinct segments is developed in the next chapter. Our interest here is simply to show that the number grows so rapidly that explicitly encoding an analytic

form for each one is not a practical proposition). The triplets at either end are the qualitative values of the function, its first and second derivatives and the sign in the centre is the qualitative value of the second derivative over the interval.

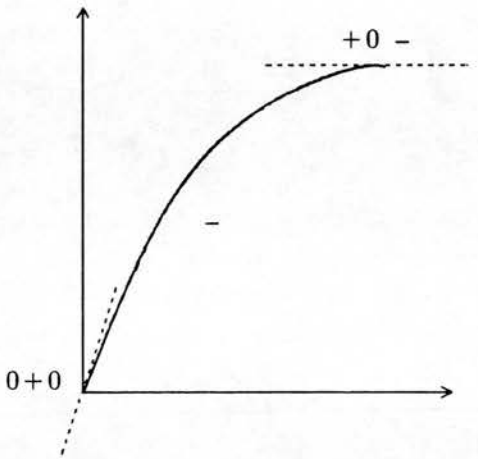


Figure 3-10. A Curve Beginning With arc3

Eighteen unreflected distinct types of curve segment are known in all (listed in the next chapter). The table below shows the number of qualitatively distinct behaviours that can be made by extending `arc3` by concatenating segments in all legal ways such that the final behaviour contains n critical points.

Initial Segment	No. of Critical Points	No. of Behaviours
`arc3`	1	6
`arc3`	2	17
`arc3`	3	91
`arc3`	4	423

Table 3-1. Library Growth as Number of Critical Points Increase

Bearing in mind that there are x, y and xy reflections of these segments then (removing 5 indistinguishables and 32 segments with asymptotes on the left) this means there are 35 distinct segments (of which `arc3` is but one) as possible initial states for some behaviour.

Clearly, we will quickly be swamped by the number of possibilities. Hence we need an alternative strategy for dealing with behaviours containing more than 1 or 2 critical points.

3.7.3 Library Index

As the maximum number of critical points tolerated increases the library grows exponentially. We do not want to have to search the entire library whenever we attempt to find a trial function congruent to some behaviour. Therefore, an indexing scheme is required. A simple index is the name of the segments within a behaviour taken in sequence from the initial state. If the library is arranged as a dictionary indexed over segment sequences we can efficiently retrieve the appropriate analytic hypothesis corresponding to some behaviour. The only complication is that **QSIM** output is not sufficiently well specified to guarantee a unique segment interpretation for a given snippet of qualitative behaviour (*e.g.* **QSIM** does not distinguish between asymptotic and direct approach). Therefore a particular **QSIM** behaviour could correspond to several segment-sequence interpretations..

3.7.4 Library Entries

For each library entry we require certain information relating its analytic and qualitative forms and the substitutions to be made for various collocation fractions. Each library entry is therefore an object whose attributes include:-

name:	the conjecture's name
function:	the defining functional form
descriptor:	the sequence of pairs of qualitative states of the function and its first derivative
consequences:	a set of relations and predicates this conjecture entails
parameters:	the adjustable parameters in the conjecture

For efficiency, other information *e.g.* derivatives and the inverse function are also stored.

3.7.5 Pragmatics

Clearly the effort required to extend the library is considerable. However, it is a one-off enterprise and could conceivably be automated. Nevertheless it is most unlikely to be a realistic proposition in general. The only advantage of completeness modulo n principle is that it gives us a way of deciding what functions to store in the library for low values of n. This is rather better than the argument proffered by Acton & Squire.

In any case, we must remember that the motivation behind **analytic abduction** was to find approximate solutions that were both *comprehensible* (their qualitative properties should be easily derived) and *adequate* (they should fit the differential equation sufficiently well and yield relationships of practical use). If the number of critical points starts to become too large and are trial functions too elaborate then comprehensibility is diminished.

An alternative approach in such cases would be to attempt to simplify the problem in such a way that the complexity of the solution is reduced *e.g.* by approximating a coefficient with a simpler function (perhaps even a constant). Alternatively one could impose order of magnitude assertions on the qualitative behaviour prediction to assert that some feature is dominant over another. This could allow us to reduce the number of critical points as a graphical example shows.

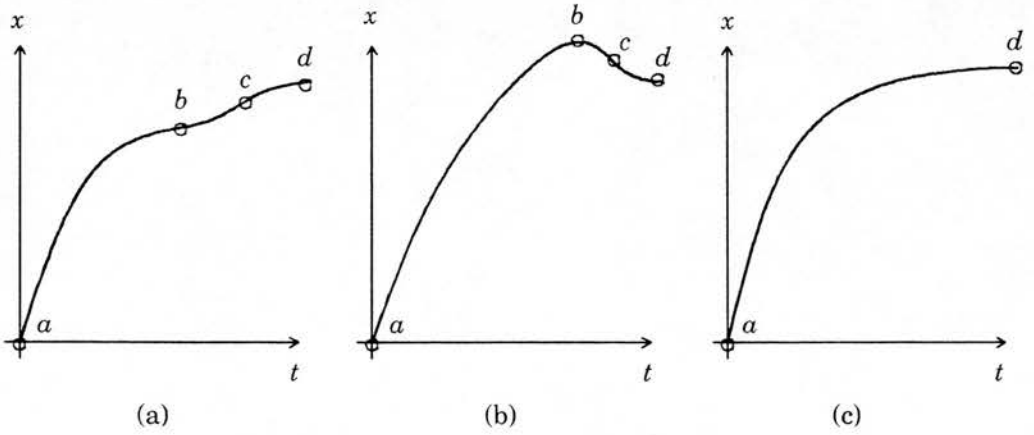


Figure 3-11. Critical Point Suppression

Figures (a) and (b) contain 4 critical points (*a* through *d*) but could conceivably be "smoothed" under various order of magnitude assertions to curve (c) (which only contains 2

critical points). However, we do not as yet have a theory of how such coercions may be applied in a controlled way so the idea remains speculative.

3.8 Limitations

The current version of **analytic abduction** is handicapped in a number of ways. First our reliance on **QSIM** to predict the qualitative features of a solution restricts us to ordinary differential equations whose coefficients are free of the independent variable. This means, for example, that we cannot reason about forced oscillators. Moreover, we inherit the problem of spurious behaviours so that we cannot be sure, if **QSIM** predicts multiple behaviours, that they all correspond to real solutions. Future versions of **QSIM** will hopefully improve the situation. Already Kuipers has added timescale abstraction [Kuipers 87], higher order derivative constraints [Kuipers & Chiu 87], trajectory non-intersection constraints [Lee & Kuipers 88] and partial quantitative knowledge [Berleant & Kuipers 88] to limit the spurious behaviours.

Second, the current library is only complete modulo 1. If the qualitative behaviour of a solution contains more than 2 critical points (*i.e.* zeroes of any parameter) then we will not necessarily be able to abduce an analytic function *congruent* to it. In such cases we have two options: either apply **segment calculus** to attempt to find a composite function which is congruent to the qualitative behavioural prediction or impose order of magnitude assumptions regarding the relative significance of critical points so that some may be suppressed. The first approach works because combining known library entries each containing n critical points (or less) under mathematical operations such as addition, multiplication or composition, can induce a behaviour with more than n critical points. The second, because suppressing critical points can allow us to smooth a behaviour to one recognisable in the library.

A third limitation is that the mapping between qualitative behaviours and known functions is one to many. This leads to inefficiencies (if the first function we abduce is not *adequate*) or

failure (if none of the topologically equivalents forms are *adequate*). We invented the exaggeration technique to ameliorate the latter problem but it is still not possible to guarantee success.

Finally, because we require *ordinary* differential equations to perform parameter optimization we cannot treat systems which are only partially specified *i.e.* where functional relationships may only be known to be monotonic. By contrast, **QSIM** is able to reason from *qualitative* differential equations (either specified directly or derived from ordinary differential equations). However, for those systems which are completely specified **analytic abduction** is capable of making more precise inferences than **QSIM**. Moreover, in the real world many engineers would guess some kind of approximate functional form for a monotonic relation and explore its ramifications rather than attempt to reason with it purely on its monotonicity properties. Indeed, this is an integral part of mathematical modelling.

3.9 Conclusions

We have described **analytic abduction**, a computational technique for mapping qualitative behaviours back to closed form mathematical expressions. Our aim was most definitely not to guess the "right" solution. Instead we sought an approximate solution which was both *congruent* to the qualitative behaviour of the system, *adequate* with respect to its differential equation, and exposed the coarse relationships between problem parameters. This allowed us to derive a time scale for a process which is something no qualitative reasoning system currently does.

However, because this requires an abduction, such inferences are not uniquely determined. Their usefulness rests on providing a concise description of behaviour and highlighting the relationship between significant system parameters. Each stage in the procedure attempted to capture the modes of inference that a human engineer would use. The optimisation and

applicability tests were based on existing engineering principles but we believe ours was the first system to exploit them computationally.

Analytic abduction differs from traditional weighted residual minimization methods by preferentially seeking single term approximations rather than sums of basis functions. Thus if we are able to find an approximation it is generally much easier to comprehend.

Moreover, we improved upon Acton & Squire's hand-cranked approach [Acton & Squire 85] by automating the envisionment of the qualitative properties of the solution and extending the theory with *topological isomorphism* and *exaggeration* to ameliorate cases where the first trial function proved to be inadequate.

We also proposed two methods for overcoming the problem of having to encode a separate analytic function for each qualitatively distinct behaviour. The first was to explain an unrecognised behaviour as the interaction of two known functions under a mathematical operation (in general this can introduce more critical points than are present in either function). This idea is developed in the next chapter. The second suggested using order of magnitude knowledge to "smooth" a qualitative behaviour reducing the number of critical points present sufficiently to recognise a function.

The essential idea is that pruning away less important details can facilitate comprehension and rough calculation. This may be useful in the early stages of design or tutorial exposition and is in any case an interesting formalization of the modes of reasoning engineers appear to use.

A possible extension of **analytic abduction** would be to modify the way we match qualitative behaviours to known functions to avoid exclusively matching sinusoidal conjectures to oscillations and also to reason about possible relative orders of magnitudes amongst the landmark values.

Many real world dynamic systems are far too complex to be solved exactly by mathematical analysis. Moreover, even if they were soluble they would often result in highly complicated formulae which defy intuition. For some applications, *e.g.* in engineering or tutoring systems, it would be appropriate to sacrifice mathematical precision for physical comprehensibility. Qualitative simulation could be used to solve abstractions of the original

problem and analytic abduction then to map the predictions back to mathematical expressions. The methods described above are a step towards achieving this objective.

Chapter 4

Segment Calculus

4.1 Introduction

Analytic abduction is a technique for conjecturing a closed form function given a qualitative description of it. However, as it currently stands, this is impossible if there is no record relating the particular sequence of qualitative states to a function in the standard library. **Segment calculus** is designed to lessen this deficiency.

Given a qualitative description of a function as a sequence of qualitative states, **segment calculus** is able to explain this as the interaction of two other qualitative functions, under an operator (*product, sum, composition* or *exponentiation*), whose mathematical forms may then be conjectured using **analytic abduction**. Apart from functional decomposition into orthogonal series this is an unsolved problem of mathematics and would have very general applicability in many fields. As in the previous chapter, the fact that we are abducing from qualitative data means we cannot guarantee that the functions we map to are the correct mathematical description. However, if we had additional constraints (*e.g.* that the function must satisfy some differential equation) then the techniques of **analytic abduction** could again be used to tune the function with respect to the equation.

Note, however, that at the very least, **segment calculus** will suggest a *functional* form which may be used to guide equation parsing techniques (see *Chapter 5*).

4.2 Overview

We begin with the observation that any qualitative behaviour can be built from (and hence partitioned into) a finite number of monotone curve segments (the complete unreflected set is listed in §4.4). Our goal is to explain the observed qualitative behaviour as a combination of two simpler behaviours under *multiplication, addition, composition* or *exponentiation*. A damped oscillation, for example, could be explained as a constant amplitude oscillation multiplied by an exponentially declining term. This has two advantages: first we need not store as many functions in the **analytic abduction** standard library. Second, we might gain some insight into processes at work in a model by separating a behaviour from a trend.

In order to explain a whole behaviour, it is partitioned into its component monotone segments, each is explained separately and then the partial results are stitched together. Each separate explanation consists of a left segment, s_i^k, an operator, **R**, and a right segment, s_j^k. The intended interpretation is that the observed segment can arise from the interaction of segments s_i^k and s_j^k under **R**. Once all possible explanations of every component of the input behaviour have been found, an explanation spanning the whole input behaviour can be built by concatenating members of consecutive left explanations and consecutive right explanations in all legal ways. Eventually this process finds all possible *spanning interpretations*.

As the output from **QSIM** will be a sequence, S, of $2n+1$ qualitative states alternately at and between time points, S can be partitioned into n monotone segments (*m-segments*) $s^1,s^2,...s^n$ whose end point values coincide successively with the $n+1$ time points. Therefore, S is composed of a concatenated sequence of s^k,

$$S = s^1 \parallel s^2 \parallel .... \parallel s^n$$

where || denotes concatenation.

For each s^k let s_i^k and s_j^k be pairs of segments and **R** an operation such that $s^k = s_i^k \ \mathbf{R} \ s_j^k$ ("=" denotes qualitative equality). The task is to find a sequence of s_i^k and s_j^k ($k=1,2,...n$) such that the s_i^k's and s_j^k's independently concatenate to form two qualitative functions with the concatenation rule

$$(s_i^k \ \mathbf{R} \ s_j^k) \parallel (s_i^{k+1} \ \mathbf{R} \ s_j^{k+1}) = (s_i^k \parallel s_i^{k+1}) \ \mathbf{R} \ (s_j^k \parallel s_j^{k+1})$$

holding for each k. Such a sequence of s_i^k and s_j^k is said to *span* S. Intuitively, a possible explanation for S can be constructed from the interaction of two qualitative functions I and J under **R** by chopping it into pieces, explaining each piece as the interaction of two segments under **R** and stitching the individual segments together to make I and J.

In the simplest case, the sequences of s_i^k and s_j^k each concatenate to monotone functions. Where they do not, concatenation is only allowed if a function, having the sequence of states seen so far, is already known (for justification see § 4.6.1).

4.3 Issues

4.3.1 Ontology

The first question which must be addressed is what constitutes the *minimal* set of segments such that any continuously differentiable function can be built from, or partitioned into, a sequence of elements from this set? As our intended application involves using qualitative properties to invoke analytic functions we shall want to be able to differentiate between direct and asymptotic approach and finite and infinite values in addition to purely sign information.

4.3.2 Representation

Second, what features of segments are important to facilitate prediction of their interactions and how are these to be represented? In particular, is there enough information in just the signs of quantities to adequately constrain prediction?

4.3.3 Decomposition Procedure

Third, how can a qualitative behaviour be decomposed into a sequence of primitive segments? This can be ambiguous due to the use of a qualitative representation of time which does not distinguish between the final time point being finite or infinite. Hence it is generally unclear whether a final value is approached directly, in finite time, or asymptotically with the final value being "attained" *at* infinite time.

4.3.4 Local Interpretations

Fourth, given a qualitative behaviour and the task of explaining it as the interaction of two other qualitative behaviours under some mathematical operation, how can each of its constituent segments be first explained as the interaction of two others in all possible ways? These will form the set of *local interpretations* from which the *spanning interpretations* must be built.

4.3.5 Spanning Interpretations

Finally, how can paths through the set of *local interpretations* be found such that adjacent interpretations along a path fit together smoothly to form an interpretation spanning the whole input qualitative behaviour? The construction of such *spanning interpretations* is the goal of **segment calculus** as they explain how the input qualitative behaviour can be obtained from a combination of two simpler qualitative behaviours.

4.4 Primitive Segments

4.4.1 Primitive m-segments

We define a set of primitive segments from which any continuously differentiable function may be constructed by concatenation or into which it may be decomposed by segmentation. We identify 18 primitives and derive 54 others by reflection in the x-,y-, and both x&y-planes.

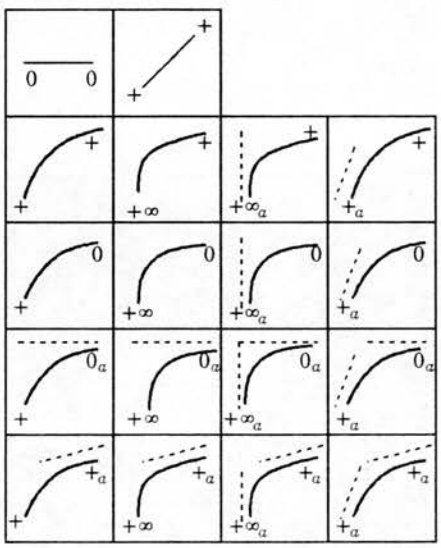


Table 4-1. Primitive m-segments

Each entry corresponds to a distinct, unreflected, segment **Type**. For example, the **Types** of the third row traversed left to right are, **arc3**, **arc3**$_i$, **arc3**$_{ia}$, **arc3**$_{fa}$ using the subscript notation "i" for infinite, "a" for asymptotic and "f" for finite.

Some reflected segments are isomorphic to others and may therefore be deleted without loss of generality.

Each end of a segment is labelled with the sign of the first derivative at that point. This serves as the principal feature on which classification is based. Consequently a segment of any given type can have multiple manifestations each corresponding to different qualitative values for the function and second derivatives at the end points.

In most qualitative reasoning systems [Bobrow 84] the three qualitative values {–,0,+} are used. **Segment calculus**, in contrast, distinguishes between ten qualitative values:

$$\{-\infty\ , -, 0, +, +\infty\ \} \cup \{-\infty\ _a, -_a, 0_a, +_a, +\infty\ _a\}$$

The first set allows for the additional distinction between finite and infinite values; the second for the distinction between direct and asymptotic approach. Whilst these differences are suppressed in other systems it is important for us to be able to distinguish between them as they correspond to very different *functional* behaviour.

However, these extra distinctions greatly increase the number of distinct curve segments and consequently the number of possible concatenations to check when building *spanning interpretations*. Fortunately, we can exploit continuous differentiability by recognising that asymptotes and infinite values may only ever appear at the boundaries. Consequently, we need only reason with the 3 qualitative values {–, 0, +} and postpone labelling the boundary triples until a spanning interpretation has been constructed. Only then need we consider that a value "0" is possibly "0_a" (asymptotically zero) and a value "+" is possibly "$+\infty$ " *etc.* This drastically reduces the size of the search space whilst retaining the expressiveness of the extended qualitative value set.

4.4.2 Representation of m-segments

An *m-segment* is represented by a `Type`, a lower bound triple and an upper bound triple.

$$\text{segment}(\texttt{Type},[f_{lb},f_{lb}',f_{lb}''],[f_{ub},f_{ub}',f_{ub}''])$$

The `Type` coincides with the family from which it is derived together with its x, y or $x\&y$ reflections. `Type` information is used critically at the concatenation stage because it enables us to determine whether one segment subsumes another. The triples denote the qualitative values of the function, its first derivative and its second derivative respectively. The description of a segment is therefore in terms of the behaviour of its end point values. The notion of convexity over the interval between end point values is contained implicitly in the segment's `Type`.

4.4.3 Reflections of m-segments

We define reflections *via* mappings between triples.

x-reflection

$$\text{segment}(\mathbf{N},[f_{lb},f_{lb}',f_{lb}''],[f_{ub},f_{ub}',f_{ub}'']) \longmapsto \text{segment}(\mathbf{N\text{-}Rx},[f_{ub},-f_{ub}',f_{ub}''],[f_{lb},-f_{lb}',f_{lb}''])$$

y-reflection

$$\text{segment}(\mathbf{N},[f_{lb},f_{lb}',f_{lb}''],[f_{ub},f_{ub}',f_{ub}'']) \longmapsto \text{segment}(\mathbf{N\text{-}Ry},[-f_{lb},-f_{lb}',-f_{lb}''],[-f_{ub},-f_{ub}',-f_{ub}''])$$

x&y-reflection

$$\text{segment}(\mathbf{N},[f_{lb},f_{lb}',f_{lb}''],[f_{ub},f_{ub}',f_{ub}'']) \longmapsto \text{segment}(\mathbf{N\text{-}Rxy},[-f_{ub},f_{ub}',-f_{ub}''],[-f_{lb},f_{lb}',-f_{lb}''])$$

Notice that under these maps $(\mathbf{Rx})(\mathbf{Ry})s = (\mathbf{Ry})(\mathbf{Rx})s$ but they differ from the usual reflections in that the direction in which a segment is traversed is always left to right regardless of reflections. However, repeated reflections of like type yield identity relations just like traditional reflections *i.e.* $(\mathbf{Rx})(\mathbf{Rx})s = s$ *etc.*

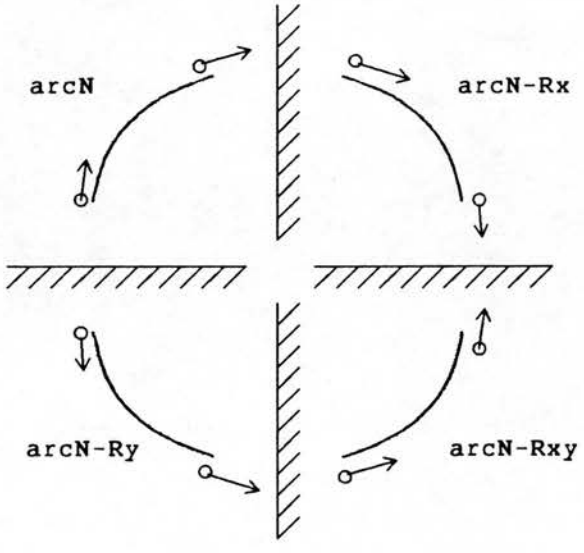


Figure 4-1. Possible m-segment Reflections

The arrows indicate both the value of the derivatives at the end point and the direction in which the segment is traversed.

4.5 Segmenting a Qualitative Behaviour

The figure below shows how a function is decomposed into a sequence of *m-segments*.

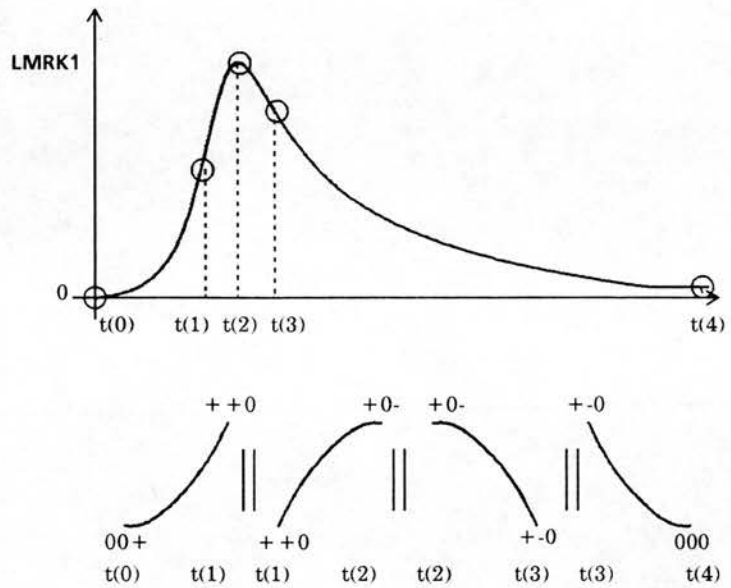


Figure 4-2. Segmenting a Qualitative Behaviour

Notice that the end points of *m-segments* coincide with the critical points (*i.e.* turning points and inflections) of the input function.

If `QSIM` had generated this function, then it would be represented as

time	function	derivative
t(0):	<0,std>	<0,inc>
t(0,1):	<[0,LMRK1],inc>	<[0,LMRK2],inc>
t(1):	<[0,LMRK1],inc>	<LMRK2,std>
t(1,2):	<[0,LMRK1],inc>	<[0,LMRK2],dec>
t(2):	<LMRK1,std>	<0,dec>
t(2,3):	<[0,LMRK1],dec>	<[LMRK3,0],dec>
t(3):	<[0,LMRK1],dec>	<LMRK3,std>
t(3,4):	<[0,LMRK1],dec>	<[LMRK3,0],inc>
t(4):	<0,std>	<0,std>

where **LMRK1** is the qualitative value of the function at the crest and **LMRK2** and **LMRK3** the qualitative values of the derivative at the inflections.

This is not the segmentation required as it is essentially a sequence of qualitative states alternately at and between time points. Instead it must be abstracted into a sequence of triple transitions across overlapping intervals. However, **QSIM** has correctly identified the critical points at which the curve must be partitioned together with the qualitative value of the convexity over each open interval. Consequently, the translation between **QSIM** output and the required segmentation is straight forward. Hence, using the notation

$$\langle f_{lb}, f'_{lb}, f''_{lb} \rangle \longmapsto_{convexity} \langle f_{ub}, f'_{ub}, f''_{ub} \rangle$$

the qualitative behavioural description becomes:

```
t[0,1]:   <0,0,+>   ⟼+   <+,+,0>
t[1,2]:   <+,+,0>   ⟼-   <+,0,->
t[2,3]:   <+,0,->   ⟼-   <+,-,0>
t[3,4]:   <+,-,0>   ⟼+   <0,0,0>
```

In fact, any continuously differentiable function (**QSIM** produces no other), can be partitioned into a sequence of *m-segments*. The sign of the convexity of any *m-segment* is constant within its interval (*i.e.* ignoring end points). This has important ramifications when dealing with periodic functions; a point to which we shall return in § 4.8).

4.6 Local Interpretations

Having segmented some qualitative function into its constituent *m-segments* the next task is to generate the set of *local interpretations* under which each *m-segment* can arise as the interaction of two other segments under some operation **R**. Having done this for all input segments, we construct a *spanning interpretation* which stitches the individual *local interpretations* together in all legal ways.

Formally, a *local interpretation* of a segment is a map, I_L, which takes a segment to a pair of segments, an operation **R** and a set of order of magnitude assumptions, **O**:

$$I_L: s^k \longmapsto s_i^k \times \mathbf{R} \times s_j^k \times \mathbf{O}.$$

The constraints on the triples ensure that the qualitative signs of the lower bound and upper bound values preserve the integrity of the equations defining the function, the first derivative and the second derivative. For example if s^k, s_i^k and s_j^k are defined by

$$s^k = \text{segment}(\mathbf{T},[h_{lb},h_{lb}',h_{lb}''],[h_{ub},h_{ub}',h_{ub}''])$$

$$s_i^k = \text{segment}(\mathbf{T}_i,[f_{lb},f_{lb}',f_{lb}''],[f_{ub},f_{ub}',f_{ub}''])$$

$$s_j^k = \text{segment}(\mathbf{T}_j,[g_{lb},g_{lb}',g_{lb}''],[g_{ub},g_{ub}',g_{ub}''])$$

then the following qualitative constraints are imposed

$$h_{lb} = f_{lb}\,\mathbf{R}\,g_{lb}$$

$$h_{lb}' = (f_{lb}\,\mathbf{R}\,g_{lb})'$$

$$h_{lb}'' = (f_{lb}\,\mathbf{R}\,g_{lb})''$$

together with a similar set for the upper bound. The precise form of the right hand terms depends on the choice of **R**. The complete set for **R** being *addition*, *multiplication*, *composition* or *exponentiation* is listed in Table 4-2 below.

h	h'	h''
$f+g$	$f' + g'$	$f'' + g''$
$f * g$	$f' * g + g' * f$	$f'' * g + 2\,f' * g' + g'' * f$
$f \circ g$	$f' * g'$	$f'' * g' * g' + f' * g''$
e^f	$e^f * f'$	$e^f * f' * f' + e^f * f''$

Table 4-2. Derivative Definitions for Various Operators

N.B. for the composition case, $f \circ g$, f is a function of g so f' is differentiation with respect to g. In all the other cases the primes denote differentiation with respect to time. The general exponential case of $h = f^g$ leads to a combinatorial explosion of possible signs and magnitude orderings so we settle for the simpler case of a constant base. In principle this generalizes to any operator by simply applying the rules of differentiation to obtain the requisite constraint equations.

The purpose of these constraints is to allow us to define a notion of qualitative equivalence between a segment and an interpretation *via* consistency of the three sets of triples at each end.

Formally, a segment is qualitatively equivalent to an interpretation if and only if the upper and lower bound triples of the segment are precisely those computed by the interpretation and the interior *convexities* are consistent with qualitative arithmetic *i.e.*

$$\text{lb-triple}(s^k) = \text{lb-triple}(s_i^k) \mathbf{R} \text{ lb-triple}(s_j^k) \wedge$$

$$\text{ub-triple}(s^k) = \text{ub-triple}(s_i^k) \mathbf{R} \text{ ub-triple}(s_j^k)$$

The order of magnitude assumptions, **O**, are essential to distinguish the context under which a sign has been computed. Table (a) below shows standard qualitative addition [*e.g.* de Kleer & Brown 84] whilst table (b) shows qualitative addition with *dynamic contexts i.e.* the ability to assert a magnitude ordering assumption to disambiguate the calculation of a sign.

[X] ⊕ [Y]		[Y] +	[Y] 0	[Y] −
[X]	+	+	+	?
	0	+	0	−
	−	?	−	−

(a)

[X] ⊕ [Y]		[Y] +	[Y] 0	[Y] −
[X]	+	+	+	+ if $\|X\|>\|Y\|$ 0 if $\|X\|=\|Y\|$ − if $\|X\|<\|Y\|$
	0	+	0	−
	−	− if $\|X\|>\|Y\|$ 0 if $\|X\|=\|Y\|$ + if $\|X\|<\|Y\|$	−	−

(b)

Table 4-3. Qualitative Addition

For example, in computing the possible sign of the second derivative of a product $h = f*g$, it is necessary to compute the possible sign of the sum

$$h'' = f''*g + 2*f'*g' + g''*f$$

If a branch was being explored for which it was already known that $f=-$, $g=+$, $f'=+$, $g'=-$, $f''=+$, and $g''=+$ then any of the three possibilities $\{-,0,+\}$ would be possible under different order of magnitude assumptions about the terms in the equation.

$$\begin{array}{lcll} h'' = & + & \text{if} & |g''*f| < |f''*g + 2*f'*g'| \quad \wedge \quad |2*f'*g'| < |f''*g| \\ h'' = & 0 & \text{if} & |f''*g + 2*f'*g'| = |g''*f| \quad \wedge \quad |2*f'*g'| < |f''*g| \\ h'' = & - & \text{if} & |f''*g + 2*f'*g'| < |g''*f| \quad \wedge \quad |2*f'*g'| < |f''*g| \\ & & & |f''*g + 2*f'*g'| < |g''*f| \quad \wedge \quad |2*f'*g'| = |f''*g| \\ & & & |f''*g + 2*f'*g'| < |g''*f| \quad \wedge \quad |2*f'*g'| < |f''*g| \end{array}$$

Hence, it is necessary to include the order of magnitude context under which a sign has been computed in forming an interpretation. Within an interpretation we retain the order of magnitude assumptions necessary to compute both the lower bound triple and the upper bound triple as

ordmgtde(`AssumpForLowerBound`, `AssumpForUpperBound`)

A final point to note is that the end points of the *m-segments* involved in the interpretation and the end points of the primitive segment being explained all coincide.

4.7 Spanning Interpretations

Having obtained all *local interpretations* of each *m-segment* comprising the input function, the process of building *spanning interpretations* may begin. This is accomplished by testing whether two adjacent *local interpretations* can concatenate to form a new one spanning both intervals. When repeated, this eventually yields an interpretation spanning the whole set of time intervals: a *spanning interpretation.*

Two *local interpretations* concatenate according to the following rule:

$$\{s_i^k \,\mathbf{R}\, s_j^k, \text{ordmgtde}(L^k,U^k)\} \parallel \{s_i^{k+1} \,\mathbf{R}\, s_j^{k+1}, \text{ordmgtde}(L^{k+1},U^{k+1})\}$$

$$= \{(s_i^k \parallel s_i^{k+1}) \;\mathbf{R}\, (s_j^k \parallel s_j^{k+1}), \text{ordmgtde}(L^k,U^{k+1})\}$$

subject to three conditions being respected

1) type consistency

2) qualitative sign consistency

3) qualitative magnitude consistency

4.7.1 Type Consistency

We define a subsumption relation over types $\supset$ to mean that a segment of one type is contained within another *e.g.* `arc3` $\supset$ `arc2` means that `arc2` is contained within `arc3`.

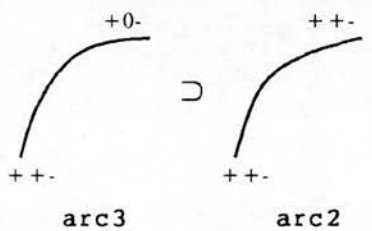


Figure 4-3. Segment Subsumption

Then we can say two segments are type consistent under concatenation if and only if

$$\text{type}(s_i^k) = \text{type}(s_i^{k+1}) \text{ or}$$

$$\text{type}(s_i^k) \supset \text{type}(s_i^{k+1}) \text{ or}$$

$$\text{type}(s_i^k) \subset \text{type}(s_i^{k+1}) \text{ or}$$

$$\exists\, fn\text{: adjacent}(s_i^k, s_i^{k+1}, fn)$$

the last possibility allows non-monotonic functions to be built from *m-segments* provided we have knowledge of some particular function having the sequence of *m-segments* seen so far. We *have* to restrict this to only those functions we know about otherwise *every* m-segment sequence would be permitted and the system would have no inferential power!

4.7.2 Qualitative Sign Consistency

Two segments are sign consistent under concatenation if and only if

$$\text{ub-triple}(s_i^k) = \text{lb-triple}(s_i^{k+1})$$

i.e. they share the same triple of signs for function, first derivative and second derivative at the point at which they are joined.

4.7.3 Qualitative Magnitude Consistency

At the join, the upper bound relative magnitude assumptions of one interpretation meet with the lower bound relative magnitude assumptions of the other. Our original motivation for the inclusion of explicit magnitude information was to check, when concatenating segments, that their adjoining ends shared consistent magnitude assumptions; the rationale being that it would then be impossible to mix behaviours.

However, the order of magnitude distinctions made to disambiguate the context under which a sign was computed proved to be expensive to maintain. If there are n ways of computing a particular triple of qualitative values for a fixed labelling of parameters under different relative magnitude assumptions this means there will be n segment pairs which are only distinguishable by their magnitude context. To retain each one as a distinct interpretation quickly led to a combinatorial explosion of possible interpretations.

Our solution to this was to filter the spanning interactions at each stage to eliminate "duplicates" by merging all interpretations having the same mid-point triple of qualitative values, regardless of their magnitude context. Unfortunately, this tactic obviated the need for performing explicit magnitude consistency filtering in the first place. Think of it as follows: the upper bound triple of one segment is guaranteed to be identical to the lower bound triple of the other and the same equations define the first and second derivatives at both ends. Consequently, there is bound to be a pair of identical contexts for computing a particular sign for any expression. As we then proceed to discard the magnitude information at the join any discriminatory advantage gained by computing the context

information is lost. Hence there is no reason to check for context consistency in the first place.

There is however, one advantage left in all this. As the end points of the spanning interpretations do not join on to other segments, their magnitude contexts are never suppressed. Consequently, if an analytic function was sought which was to match the qualitative behaviour corresponding to each part of a *spanning interpretation*, the end-point magnitude contexts could be used to discriminate between analytic functions whose qualitative behaviours are, in all other respects, equivalent. We therefore, retained magnitude contexts in the segment representation.

If each of the three above criteria can be satisfied in some way then the concatenation of two interpretations is permitted.

The next question that must be addressed is whether or not this formalism is sufficient? In other words, is it possible to decompose any input function into interactions of *m-segments* whose end points coincide with the critical points of the input function? The answer unfortunately is no. Problems arise when there is a phase mismatch between the input function and those we would want it to be decomposed in terms of. To demonstrate this we need to consider the properties of periodic functions.

4.8 Periodic Functions

To understand the problems periodicity introduces, it is necessary to consider how qualitative periodic functions can be represented. To do this we will abstract the features an analytic representation must have. Rather than recite the mathematical properties in isolation it is more germane to examine how Sacks represented periodicity in his **QMR** system [Sacks 85b]

4.8.1 Features of Analytic Periodicity

Sacks' central thesis is that where possible qualitative reasoning is to be performed by solving a mathematical specification of the system and deriving the qualitative behaviour of the analytic solution. This led him to problems concerning periodic motion and hence to the question of representing such behaviour in a form of use to the computer. His solution was to represent any piecewise continuous function by a sequence of *fun-ints* defined over closed interval between [*lb*,*ub*]. A function was strictly monotonic or constant over the range of one *fun-int*. Each *fun-int* was essentially a frame having slots for *lb*, *ub*, $f(t)$, $f(lb)$, $f(ub)$, $f'(t)$, $f''(t)$, *directionality*, *convexity*, $f^{-1}(t)$ *etc* all represented in explicit analytic terms. Periodic motion could be represented by a finite number of *parameterised fun-ints*. For example, Sacks would represent $\cos(t)$ as follows

Property	**Value**	
direction	down	up
lb	$2n\pi$	$(2n+1)\pi$
ub	$(2n+1)\pi$	$2(n+1)\pi$
fun	$\cos(t)$	$\cos(t)$
lb-val	1	−1
ub-val	−1	1
inverse	$\arccos(t)$	$\arccos(t)$
derivative	$-\sin(t)$	$-\sin(t)$
der2	$-\cos(t)$	$-\cos(t)$
convexity	$(\ 2n\pi\ \ -1\ \ (2n+1)\pi/2\ )$ $(\ (2n+1)\pi/2\ \ +1\ \ (2n+1)\pi)$	$(\ (2n+1)\pi\ \ +1\ \ 2(n+1)\pi/2\ )$ $(\ 2(n+1)\pi/2\ \ -1\ \ 2(n+1)\pi)$

Table 4-4. QMR's Representation of Cosine

with n = ...,−2,−1,0,1,2,...

Two crucial features of this representation are the use of exact analytic expressions and quantitative time points. This allows Sacks to define a parameterised fun-int to be periodic if

1) the **ub**, **singularities** and **convexity** entries are a fixed distance, independent of n, from the **lb**, and

2) the **fun** satisfies $f(t)\{n/k\} = f(t)\{n/k+1\}$, $f(t)\{n/k\} < f(t)\{n/k+1\}$ or $f(t)\{n/k\} > f(t)\{n/k+1\}$ independently of k and t (where the bracketed expressions mean substitute k or $k+1$ for n in $f(t)$).

1) guarantees the curve shape repeats for all values of n and 2) that the function's values are scaled uniformly each period. In fact, because Sacks uses explicit mathematical expressions 2) can be strengthened considerably and in a later example for damped oscillation he cites the ratio of successive maxima together with the functions limit as $t \to \infty$. This concise notation allows a periodic function to be represented in finite terms. However, it is not immediately appropriate for the description of qualitative periodicity. This is because qualitative functions have neither an absolute quantitative measure of time, merely a total ordering on time points, nor analytic expressions to ensure the maxima are scaled consistently. Consequently, we can only define a weaker notion of periodicity.

4.8.2 Features of Qualitative Periodicity

The features of qualitative periodicity can be obtained as natural abstractions of Sacks' quantitative representation. First, a few definitions are necessary.

A *behaviour* is a sequence of qualitative states.

A qualitative state is defined to be *re-entrant* if it has the same sign of landmark value and same directions of change of all higher derivatives as some previous qualitative state in the behaviour.

A behaviour is then qualitatively periodic if

1) the *re-entrant* states are always separated by the same number of time points (but because they could represent different actual time intervals this is weaker than Sacks)

2) the qualitative values of a sequence of *re-entrant* states are totally ordered (but because this does not imply a unique functional relationship between maxima this too is weaker than Sacks)

Finally, an option for a convergent qualitatively periodic function would be

3) to state the limit of the qualitative value of the function as the number of time points tends to infinity

Kuipers' **QSIM** program [Kuipers 85, Kuipers 86a] does not entirely fulfil these desiderata: largely because it arrives at its behavioural predictions *via* qualitative simulation (which for a periodic function of changing amplitude is non-terminating). However, Weld [Weld 85, Weld 86] has shown how to aggregate behaviours to detect cycles and **QSIM** at least recognises a constant amplitude cycle by detecting whenever a state is attained which has been previously encountered. In our reconstructed version, detection of *re-entrant* states is a heuristic extension of **QSIM**'s cyclicity termination condition to catch damped or growing oscillatory behaviours. This is not ideal and we regard it as no more than a working convenience waiting for a rigorous solution. The integration of Weld's technique within **QSIM** would be better. Another approach would be to perform some kind of inductive proof over evolving behaviours or to reflect the analysis back to the mathematical level and attempt to infer cyclic trends directly from the form of the ordinary differential equation. Indeed the latter observation again emphasises the utility in mixing qualitative with analytic reasoning.

4.9 How Periodicity relates to Segment Calculus

Now we have some notion of what form a qualitative periodic function must have we ask a simple question: can a qualitatively periodic function be explained as the interaction of two other qualitative functions using the rules of **segment calculus** developed thus far? The answer regrettably is no.

First note that it is impossible to generate a periodic function from the interaction of two *m-segments*: one of the functions at least must be an oscillator (*i.e.* a sequence of non-subsuming *m-segments*).

The problem arises from the implicit assumption that each segment in the decomposition of the input function can be explained as the interaction of two *m-segments whose end points coincide with those of the input segment.* This is not always the case, as the phase of the input function may differ from that of the oscillator into which the input is being decomposed. A simple example will suffice.

Suppose the input function were $e^{-t}\cos(t)$ (although the functional form would not, of course, be known) and the system was seeking its decomposition into a product of two functions, e^{-t} and $\cos(t)$. Below we sketch the three functions e^{-t}, $\cos(t)$ and $e^{-t}\cos(t)$ over one period up to *re-entrant* states.

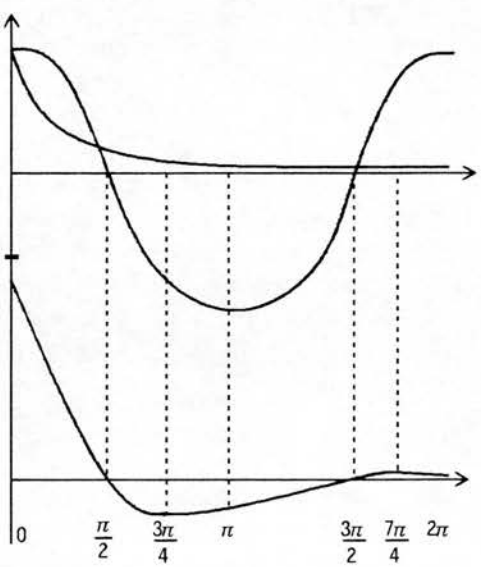


Figure 4-4. Phase Mismatch Induced by Segment Interaction

The critical feature to note is that although the zeros of $e^{-t}\cos(t)$ and $\cos(t)$ coincide, the maxima do not. Consequently, when $e^{-t}\cos(t)$ is decomposed into a sequence of concatenated *m-segments*, partitioning $\cos(t)$ at these same time points cannot split it into a sequence of concatenated *m-segments* as some segments contain inflections.

The solution we advocate is to introduce a new type of segment, *s-segments*, formed by concatenating two *m-segments* such that their join is an inflection.

4.9.1 Representation of s-segments

An *s-segment* is represented by a type and a concatenated pair of *m-segments*.

segment(**Stype**, segment(**Mtype1**,[f_{lb},f_{lb}',f_{lb}''],[f_m,f_m',f_m'']) ||

segment(**Mtype2**,[f_m,f_m',f_m''],[f_{ub},f_{ub}',f_{ub}''])

The **Stype** indicates the *m-segments* of which the *s-segment* is comprised together with its x, y or $x\&y$ reflections. The mid point triples of the two *m-segments* must be the same and the second derivative at the mid point must be zero (to guarantee an inflection). We introduce two types of *s-segment* shown in the diagram below. It is not necessary to invent *s-segments* having asymptotic or infinite values as *s-segments* are only ever used to represent oscillatory behaviours.

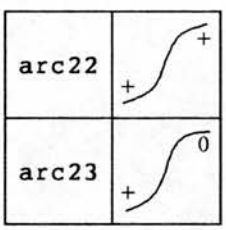


Table 4-5. Permissible s-segments

MType information is used critically at the concatenation stage because it enables us to determine whether one s-segment can possibly concatenate to an adjacent m-segment. Again, the triples denote the qualitative values of the function, its first derivative and its second derivative respectively. Similarly, the notion of convexity over the interval between end point values is contained implicitly in the segments **SType**.

4.9.2 Reflections of s-segments

S-segment reflections are defined in terms of those of the two *m-segments*.

x-reflection

$$\text{segment}(\mathbf{N}, s_{i1} \parallel s_{i2}) \rightarrow \text{segment}(\mathbf{N\text{-}Rx}, (\mathbf{Rx})s_{i2} \parallel (\mathbf{Rx})s_{i1})$$

y-reflection

$$\text{segment}(\mathbf{N}, s_{i1} \parallel s_{i2}) \rightarrow \text{segment}(\mathbf{N\text{-}Ry}, (\mathbf{Ry})s_{i1} \parallel (\mathbf{Ry})s_{i2})$$

x&y-reflection

$$\text{segment}(\mathbf{N}, s_{i1} \parallel s_{i2}) \rightarrow \text{segment}(\mathbf{N\text{-}Rxy}, (\mathbf{Rxy})s_{i2} \parallel (\mathbf{Rxy})s_{i1})$$

Note the ordering switch on *m-segments* when the *s-segment* is *x*-reflected.

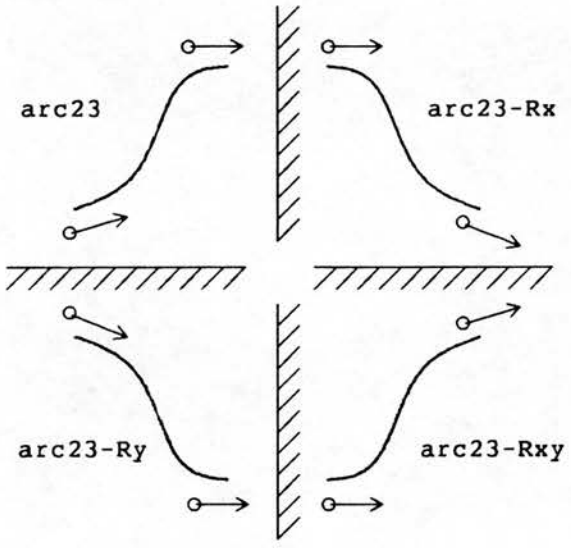


Figure 4-5. Possible s-segment Reflections

As the convexity of the two *m-segments* contained in an *s-segment* must (by definition) differ, at least one of the *m-segment* types will be a reflected type. Consequently, another modification necessary when reflecting *s-segments* is the deletion of identity reflections *e.g.* $(\mathbf{Rx})(\mathbf{Rx})s = s$ *etc.* from the *m-segment* types in forming the new **Mtype** names.

4.9.3 Deriving the Interaction Properties of s-segments

Input functions are always segmented in terms of *m-segments*. Therefore, it is important to understand how *m-segments* may be derived from the interaction of an *s-segment* and an *m-segment*. As an *s-segment* is a pair of concatenated *m-segments* this can be expressed formally as

$$s^k = (s_{i1}{}^k \parallel s_{i2}{}^k) \mathbf{R}\, s_j{}^k.$$

Note that $(s_{i1}{}^k \parallel s_{i2}{}^k)$ is an *s-segment* and must be treated as a unit because the intervening inflection coincides with no significant time point of the function under analysis.

Now any *m-segment* can be expanded as a concatenation of two *m-segments* of like or subsuming type. So $s_j{}^k$ can be represented as

$$s_j{}^k = s_{j1}{}^k \parallel s_{j2}{}^k$$

where $\text{type}(s_{j1}{}^k) = \text{type}(s_{j2}{}^k)$, $\text{type}(s_{j1}{}^k) \supset \text{type}(s_{j2}{}^k)$ or $\text{type}(s_{j1}{}^k) \subset \text{type}(s_{j2}{}^k)$ and there exists some intermediary triple coinciding with the inflection point of $s_{i1}{}^k \parallel s_{i2}{}^k$ at which the ub-triple$(s_{j1}{}^k)$ = lb-triple$(s_{j2}{}^k)$ and all qualitative constraints are respected. Hence,

$$s^k = (s_{i1}{}^k \parallel s_{i2}{}^k) \mathbf{R}\, (s_{j1}{}^k \parallel s_{j2}{}^k)$$

and therefore by applying **R** distributivity over ||

$$s^k = (s_{i1}{}^k \mathbf{R}\, s_{j1}{}^k) \parallel (s_{i2}{}^k \mathbf{R}\, s_{j2}{}^k)$$

In other words, an *m-segment* can be produced from an *s-segment* and an *m-segment* under **R** if the *m-segment* can be partitioned at the *s-segment* inflection time point into two abutting *m-segments* and each of these can independently interact with one of the *m-segments* contained within the *s-segment* to produce a pair of *m-segments* which can be concatenated to form a spanning *m-segment*.

By running this rule backwards we can generate all *s-segment* x *m-segment* interactions which can possibly give rise to an particular *m-segment* interaction and hence characterise the interaction properties of *s-segments*.

4.9.4 Representational Requirements of Library Functions

The introduction of *s-segments* imposes a necessary extension to the representation of functions which are to reside in the standard library. Before, for example, the cosine function could be represented solely in terms of a sequence of *m-segments*. Now, however, it is necessary to also include a representation in terms of an alternating sequence of *m*- and *s-segments*.

m-segment representation

```
cos ::= { segment(arc3-Rx,[+,0,–],[0,–,0]),
          segment(arc3-Ry,[0,–,0],[–,0,+]),
          segment(arc3-Rxy,[–,0,+],[0,+,0]),
          segment(arc3,[0,+,0],[+,0,–]) }
```

mixed m/s-segment representation

```
cos ::= { segment(arc23-Rx, segment(arc3-Rx,[+,0,–],[0,–,0]) ||
                            segment(arc2-Ry,[0,–,0],[–,–,+])),
          segment(arc2-Ry,[–,–,+],[–,0,+]),
          segment(arc23,    segment(arc2-Rxy,[–,0,+],[–,+,0]) ||
                            segment(arc3,[–,+,0],[+,+,–])),
          segment(arc3,[+,+,–],[+,0,–]) }
```

Figures 4-6 and 4-7 shows two alternative partitions of the cosine function, one involving only *m-segments* and the other involving a combination of *m*- and *s-segments*.

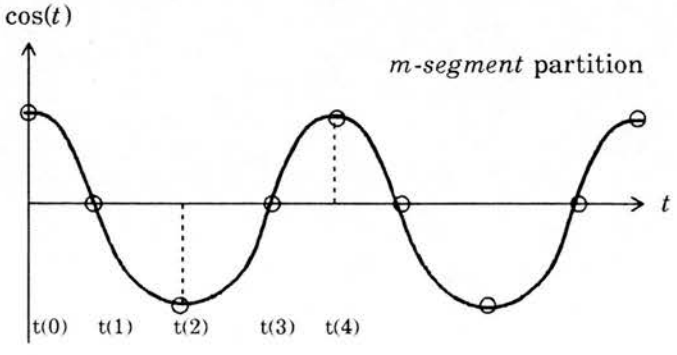


Figure 4-6. Monotone Partition of Cosine

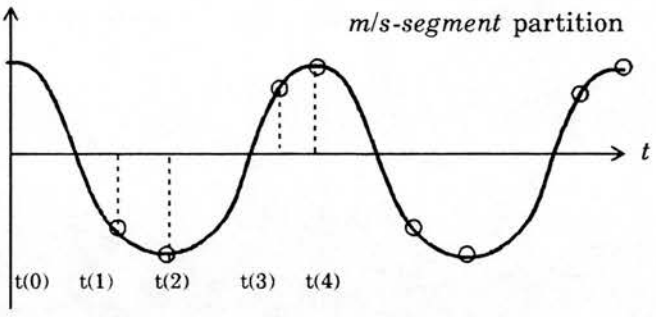


Figure 4-7. Mixed Partition of Cosine

Moreover, any cyclic permutation of the elements of the descriptor set for cosine is an equivalent representation.

4.10 Complexity

In this section we put a scale to the number of *local interpretations* of an *m-segment* represented by

$$<h_{lb},h'_{lb},h''_{lb}> \longmapsto_{convexity} <h_{ub},h'_{ub},h''_{ub}>$$

where "*convexity*" is the qualitative value of the curvature over the interval between the segment's end-points and

$$\begin{aligned} h_i &= f_i \mathbf{R} g_i \\ h'_i &= (f_i \mathbf{R} g_i)' \\ h''_i &= (f_i \mathbf{R} g_i)'' \end{aligned}$$

with $i = lb$ or ub.

To do this consider just one of the triples and the restricted qualitative value set $\{+,0,-\}$. We proceeded to count the number of ways in which each of the 27 distinct triples, $<h_i,h'_i,h''_i>$, could be obtained such that, for a given operator **R**, the conjunction of equations holds. The insistence on conjunction ensures that the qualitative values of the f_i, f'_i, f''_i, g_i, g'_i, and g''_i used to compute a particular sign for h_i, h'_i or h''_i are used consistently throughout the

equation set. This is important, as it precludes some labellings and therefore reduces the number of possibilities. The analytic form for this number will vary from case to case depending on **R**. We therefore took the expedient step of deriving these numbers by brute force one-off computation. The results are shown in Table 4-6.

Triple	Operator			
$<h,h',h''>$	$*$	$+$	$\wedge$	$\circ$
<+,+,+>	70	125	21	30
<+,+,0>	42	75	9	18
<+,+,->	78	125	9	30
<+,0,+>	38	75	9	36
<+,0,0>	26	45	3	63
<+,0,->	54	75	3	36
<+,-,+>	70	125	9	30
<+,-,0>	42	75	3	18
<+,-,->	78	125	3	30
<0,+,+>	60	75	0	30
<0,+,0>	36	45	0	18
<0,+,->	60	75	0	30
<0,0,+>	54	45	0	36
<0,0,0>	81	27	0	63
<0,0,->	54	45	0	36
<0,-,+>	60	75	0	30
<0,-,0>	36	45	0	18
<0,-,->	60	75	0	30
<-,+,+>	78	125	0	30
<-,+,0>	42	75	0	18
<-,+,->	70	125	0	30
<-,0,+>	54	75	0	36
<-,0,0>	26	45	0	63
<-,0,->	38	75	0	36
<-,-,+>	78	125	0	30
<-,-,0>	42	75	0	18
<-,-,->	70	125	0	30

Table 4-6. Number of Ways to Obtain Each Triple

For a particular operator **R** and triple $<h_i,h'_i,h''_i>$, let this number be $n(<h_i,h'_i,h''_i>, \mathbf{R})$. Given, two triples the number of legal *local interpretations*, $n(I_L)$, is bounded by

$$n(I_L) \leq n(<h_{lb},h'_{lb},h''_{lb}>, \mathbf{R}) * n(<h_{ub},h'_{ub},h''_{ub}>, \mathbf{R})$$

This is in general less than the direct product because certain pairs of triples cannot consistently coexist *e.g.* it is impossible to have a lower bound triple **<+,+,->** and an upper bound triple **<-,-,+>**. Constraints such as these, together with conditions on the *convexity*, drastically reduce the number of possibilities.

The example cited in section 4 was tested on the system seeking a *spanning interpretation* in the form of a product. The right-hand column below lists the number of possible *local interpretations* found for each *m-segment*.

t[0,1]:	**<0,0,+>**	$\longmapsto_+$	**<+,+,0>**	224
t[1,2]:	**<+,+,0>**	$\longmapsto_-$	**<+,0,->**	246
t[2,3]:	**<+,0,->**	$\longmapsto_-$	**<+,-,0>**	246
t[3,4]:	**<+,-,0>**	$\longmapsto_+$	**<0,0,0>**	200

which defines a product space of order $3x10^9$. Out of this emerged 32 candidate solutions, which in reality reduced to 8 with four fold degeneracy. For example, if $s_1 * s_2$ was a solution then so was $s_2 * s_1$ and $(\mathbf{Ry})s_1*(\mathbf{Ry})s_2$ and $(\mathbf{Ry})s_2*(\mathbf{Ry})s_1$ (*y*-reflection is equivalent to multiplication by –1). Moreover, the remaining 8 separated into two groups of 4 depending on whether the time point **t(4)** was finite or infinite. Amongst these were *spanning interpretations* which could be interpreted as the product of something like an increasing half-parabola with a decaying exponential. We cannot, of course, say it is the product of an increasing half-parabola with a decaying exponential because the mapping from qualitative to analytic descriptions is one to many.

4.10.1 One Family of Solutions

Below we give a sample of the four characteristic solutions types generated by **segment calculus**. Notice that the principal difference lies in the sign of the convexities at the extremes of the curve. The type information, is built from the primitive type name and a reflection.

```
t[0,4]: segment(arc4-Ry,[+,–,0],[0,0,0]) * segment(arc3-Rxy,[0,0,+],[+,+,0])
t[0,4]: segment(arc4-Ry,[+,–,0],[0,0,0]) * segment(arc3-Rxy,[0,0,+],[+,+,+])
t[0,4]: segment(arc4-Ry,[+,–,+],[0,0,0]) * segment(arc3-Rxy,[0,0,+],[+,+,0])
t[0,4]: segment(arc4-Ry,[+,–,+],[0,0,0]) * segment(arc3-Rxy,[0,0,+],[+,+,+])
```

4.11 Managing Complexity

The greatest improvement in efficiency is gained by avoiding dealing with the 18 distinct segment types directly. We need only reason with the 3 qualitative values {–, 0, +} and postpone labelling the boundary triples until a spanning interpretation has been constructed. Only then need we consider that a value "0" is possibly "asymptotically zero" and a value "+" is possibly "$+\infty$" *etc*. This drastically reduces the size of the search space.

Clearly there is potential for reducing the complexity of the search space by exploiting the symmetries of + and *. We saw in §4.10.1 how a four fold degeneracy arose precisely because of the symmetries of *. It would be better to generate these symmetric forms by simple algebraic manipulation from a single segment calculus interpretation rather than generating them directly within the program.

Also, instead of pre-storing the possible interpretations of single *m-segments* we could equally pre-store (the much fewer) possible *spanning interpretations* of *m-segment* sequences. We would then have to modify the segmentation algorithm to partition a qualitative function in this way.

Another possible saving could be made *via* the representation of interpretations. If we employed a tree structure indexed over pairs of segment types then when searching for *spanning interpretations*, we could exclude large numbers from further consideration at a stroke by testing for impossible concatenation on the basis of non-subsuming types.

The above techniques would all reduce the size of the search space. However, notice that all the inferences we draw are essentially propositional. Consequently, in principle, it should be possible to use a bit string representation of segments and interpretations which would significantly improve the speed of the program.

4.12 Related Work

Morgan [Morgan 87, Morgan 88] has addressed the representation of functions as a sequence of segments and defined a similar calculus over them. Although it was not his aim to tackle the problem of explaining a behaviour as the interaction of two simpler behaviours, his representation could serve this purpose too. In this scheme, a segment is a triplet of the signs of its zero-th, first and second derivatives called a qualitative vector. The shape of a segment is assumed adequately specified by just the signs of its first and second derivatives. As there are 3 possibilities for each one this makes 9 distinct curve shapes.

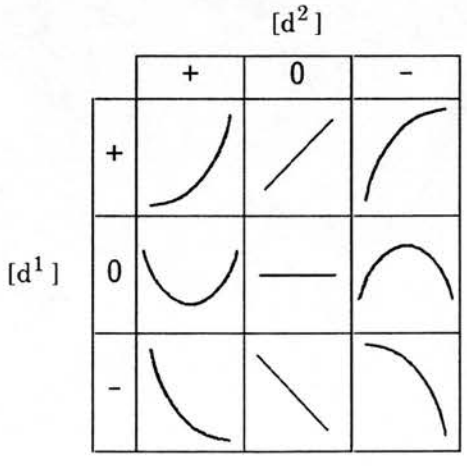


Figure 4-8. Morgan's Nine Curve Shapes

Qualitative vectors can be combined according to rules of qualitative algebra [de Kleer & Brown 84]. For example, two vectors $[a_0,a_1,a_2]$ and $[b_0,b_1,b_2]$ can be "vector multiplied" to yield $[c_0,c_1,c_2]$ where the components are direct qualitative versions of the usual derivative relationships *i.e.*

$$[c_0] = [a_0]*[b_0]$$

$$[c_1] = [a_0]*[b_1] + [a_1]*[b_0]$$

$$[c_2] = [a_0]*[b_2] + [a_1]*[b_1] + [a_2]*[b_0]$$

This is entirely analogous to our rules for combining *m-segments*.

However, there are a number of differences between our representations. Let us suppose we have a qualitative behaviour, containing n time points, generated by **QSIM**. Morgan would partition the behaviour into $2n+1$ curve shapes whilst we would partition it into n *m-segments*. Thus his curve shapes can represent a finite or infinitesimal time period and never overlap; our *m-segments* always span a finite time period and overlap precisely at the time points.

Moreover, because certain qualitative operations *e.g.* qualitative addition, can generate ambiguities Morgan allows elements of the vector to assume the value "?". For example, when we multiplied the qualitative vectors above $[c_1]$ would be indeterminate if $[a_0]=+$, $[b_1]=+$, $[a_1]=-$ and $[b_0]=+$ (similarly for other possibilities). In fact Morgan himself points out that this feature of qualitative operations will mean that the higher order derivatives (*i.e.* later elements of the qualitative vector) are more likely to have the value "?". This is something of an understatement as we have just seen how easy it is for ambiguities to creep into the *first* derivative!

Segment calculus avoids this problem by the use of *magnitude contexts i.e.* whenever an ambiguity arises we create the *context* (or relative magnitude orderings) under which a certain sign is computed and propagate the new context through subsequent constraints. Therefore, by using *magnitude contexts* signs are always computed unambiguously.

Finally, Morgan's curve shapes do not distinguish between finite and infinite gradients or direct and asymptotic approach. However, in terms of recognising functional forms this is crucial and so **segment calculus** does make these distinctions.

4.13 Conclusions

We have described a calculus of end point values and convexities for representing and reasoning about the structure of qualitative functions of the sort output from **QSIM**. Our purpose was to extend the range of problems **analytic abduction** could be used to solve by relaxing the requirement that a function having a particular qualitative behaviour is known beforehand. The current system is able to suggest products, sums, simple exponentials or compositions of functions in the standard analytic abduction library. It is possible that such a compound function would, in fact, prove to be a better approximate solution than either an almost everywhere congruent trial function or an exaggerated trial function because it would account for all the qualitative features of the exact solution without sacrificing accuracy at the initial time point.

At the very least **segment calculus** is able to provide a justification for a conjecture pertaining to the compound form of the solution of the original differential equation. This provides partial knowledge about the solution which is exploited in the **equation parsing** technique discussed in the next Chapter.

The notion of *magnitude contexts* for avoiding ambiguity by asserting enough assumptions to determine a definite qualitative sign may prove to be a useful technique in **QSIM**. However, the gains imparted by our current use of *magnitude contexts* fall short of the expectations we originally had.

Finally, it is impressive to see how few solutions actually emerge from such an enormous solution space using knowledge of the signs of quantities alone.

Chapter 5

Equation Parsing

5.1 Introduction

Segment calculus is able to suggest a qualitative *form* for the analytic solution of an ordinary differential equation. This means that the qualitative behaviour of the actual solution can be explained by the interaction of two simpler qualitative behaviours under *multiplication*, *addition*, *composition* or *exponentiation*. In the last chapter we suggested that this could then be passed to **analytic abduction** to conjecture an *approximate functional* solution which could then be tuned with respect to the original differential equation. This procedure would make use primarily of the shapes of the conjectured component behaviours. However, an alternative strategy is possible based on the proposed operator decomposition of the solution.

The hypothesis that a behaviour is, for example, the *product* of two other behaviours provides partial knowledge of the structure of the solution (the *form*). Often on substitution into the differential equation the resulting terms can be partitioned in such a way that groups can be shown to cancel out if the unknown functions are labelled a certain way. Our goal is to find a partitioning of the terms in the equation such that the interpretations of the

function symbols are *consistent* (each function symbol has a unique interpretation) and *complete* (there are no unaccounted-for terms). We call this technique **equation parsing**.

If **equation parsing** succeeds, it will reveal an *exact functional* solution of the differential equation. In this respect it is different from the other techniques developed in this thesis which all construct *approximate* solutions. However, there are two reasons why this diversion is warranted.

First, **equation parsing**, like **analytic abduction**, focuses on modelling how a mathematician might perceive an equation and how such insight can be used to facilitate solution. In **analytic abduction** we modelled the perception of the qualitative properties of the solution and used these to conjecture an approximate functional form. In the current chapter we aim to model the perception of the competing tensions within an equation and use these to guide subsequent solution. Our hope is that the insight we gain from modelling fairly simple equations might carry over to those equations which are not solvable by any standard technique (*e.g.* many nonlinear equations).

Second, although **segment calculus** was intended to explain how some qualitative behaviour could arise as the interaction of two simpler behaviours, its output supplies both the shapes of the functions involved and the operator used to combine them. This latter information is obtained at no extra computational expense. We wanted to see how much inferential leverage this, apparently weak, partial information heralded and **equation parsing** was the result.

We are not claiming that the equations in this chapter cannot be solved using standard methods. Our interest lies in modelling the perception of an equation and capitalizing on partial information about its solution.

5.1.1 Plan

The rest of the chapter is organised as follows. §5.2 motivates the **equation parsing** technique by extracting some maxims from the behaviour of real mathematicians. §5.3

explains the representational requirements for the structures we will subsequently use. §5.4 presents the **equation parsing** algorithm and describes the operation of each of its parts. This is followed, in §5.5, by a set of examples, of increasing complexity, which demonstrate the algorithm in action. §5.6 discusses some issues the parsing approach raises and proposes possible improvements for the future. Finally, §5.7 points out the limitations of the present approach, summarizes the research contributions of the chapter and motivates the technique developed in the next chapter.

5.2 Overview of Equation Parsing

Equation parsing rests on three observations concerning the behaviour of mathematicians engaged in problem solving attempts.

5.2.1 Exploitation of Partial Information

Typically, if one were able to guess the *functional form* of the solution (*e.g.* that it should be the composition of two functions) then this will often provide just enough structure to recognise what the simpler functions might be. This is analogous to the method of separation of variables commonly used for solving boundary value problems involving partial differential equations.

The method of separation of variables involves hypothesising that the solution of the equation should be a product of functions of a single variable. Under this hypothesis, the partial differential equation typically expands to a system of ordinary differential equations which yield more easily to analysis. For example, the one dimensional heat conduction equation is

$$\partial^2 u/\partial x^2 = (1/k)*\partial u/\partial t$$

To obtain a solution which decreases exponentially with time and which satisfies the boundary conditions

$$u(0,t) = u(\ell,t) = 0, \qquad t \geq 0$$
$$u(x,0) = f(x), \qquad 0 \leq x \leq \ell$$

for some prescribed function f and constant ℓ we assume a solution of the form

$$u(x,t) = X(x)*T(t)$$

On substitution into the original equation we find

$$(1/X)*d^2X/dt^2 = (1/(k*T))*dT/dt$$

As x and t are independent variables the most plausible interpretation of this equation is that both sides evaluate to one and the same constant, $-\omega^2$ say. Under this assumption, the original partial differential equation separates into two simultaneous ordinary differential equations which may be integrated by standard techniques to yield

$$X = A\cos(\omega x) + B\sin(\omega x)$$

and

$$T = C\exp(-\omega^2 kt).$$

We can solve the original problem by piecing together these partial solutions. Hence

$$u(x,t) = (D\cos(\omega x) + E\sin(\omega x))\exp(-\omega^2 kt).$$

In **equation parsing** we attempt to do something similar for ordinary differential equations. This involves extending the range of *functional forms* to candidate solutions other than simple products and considering the possible modes of termwise interaction. One of my contentions is that the only reason why humans do not do this is the sheer complexity of the interactions which ensue. However, for a computer, keeping track of these is a realistic proposition.

5.2.2 The Perception of Inverse Relations

The second observation concerning differential equation solving is that certain sub-terms in the expanded equation will pair off, combining to leave something simpler. The basic cases are $X+(-Y)$ and $X*(1/Y)$ where X and Y are syntactically dissimilar. This suggests the more abstract view that a pair of terms may be eliminated from the equation if they bear an inverse relationship to one another.

The objective of **equation parsing** is to determine the *functional form* for the solution and thence the set of termwise interactions which must consistently hold for the equation to be satisfied, together with their interpretation in terms of known analytic functions.

We can imagine the terms in the original equation being overlaid with a set of arcs signifying the inverse relations required to hold between them. Such a graph is called an *interaction graph*.

5.2.3 Solution By Inspection

In constructing the interpretation of an arc in an *interaction graph* we could treat the implied equation as a sub-problem *i.e.* a simpler differential equation to be solved. This could be done using *symbolic integration* or by *inspection*.

If we chose to use a symbolic integration program we would lay ourselves open to the charge that this could have been employed on the original problem. We would only have a defence if the original problem could not be solved using the equation solver. Even if symbolic integration were feasible only for the equations derived from arcs in the interaction graph, it does somewhat violate the spirit of **equation parsing**, which was motivated by human perception, to ultimately resort to symbolic integration.

The alternative strategy, of solving the sub-equations by *inspection*, is the one we prefer. This consists of matching each arc in the *interaction graph* to a set of known *function clichés*. I suspect (but cannot prove) that *inspection* is a reasonable model of what is taking place

inside a mathematician's head, at least in the initial stages of problem solution. Similar inspection techniques have been used to model tasks such as programming [Rich 81].

Explicit knowledge of how the function interacts with its derivatives will allow us to drive the parse either from the differential equation (top down) or from the data (bottom up). Without knowledge of clichéd behaviour the bottom up approach is impossible.

Consequently, for the purpose of the present chapter we will adopt the *inspection* approach with the rider that **equation parsing** could certainly (and fairly easily) by made more robust by grounding it in a proper symbolic integrator.

5.3 Representations

Three types of structure are employed in **equation parsing**: *differential equations* (including expanded versions), *interaction graphs* and the *function clichés*.

5.3.1 Representation of Equations

An equation is usually represented by an expression tree *i.e.* a tree whose nodes are mathematical operators and whose leaves are functions, variables or constants.

We abandon the expression tree representation of mathematical equations in favour of nested lists. This is because nested lists simplify the statement of interaction hypotheses.

5.3.1.1 Weak Normal Form

To do this, the input differential equation is first put in a *weak normal form*. This involves standardizing its operators to include only +/2, –/1, */2, *d*/*dx*/1 and ↑/2 (in the syntax

operator/arity) and sorting its terms in decreasing orders of derivatives. By placing the equation in a weak normal form we avoid having to encode equivalent matching and parsing procedures for different operators. For example, given the differential equation

$$d^2y/dx^2*x^2 + 3*(dy/dx)^3*(1/x) - (1-x^2)*y = 0$$

its weak normal form is

$$x^2*d^2y/dx^2 + 3*x^{-1}*(dy/dx)^3 + -1*(-1*x^2+1)*y = 0$$

In general the pattern into which weak normal forming places the equation is

$$f_1(x)*f_2(x)*...*d^ny/dx^n + g_1(x)*g_2(x)*...*d^{n-1}y/dx^{n-1} + ... + h_1(x)*h_2(x)*...*y + e(x) = 0$$

with the provisos that

- $X - Y$ is replaced with $X + -1*Y$
- X/Y is replaced with $X*Y \uparrow -1$
- differentials are moved to the right
- coefficients are moved to the left and ordered (">" = precedes) with

  const > *higher order poly* > *lower order poly* > *exp* > *log* > *trig* > *hyperbolic*

Weak normal forming does not modify the structure of any coefficient apart from standardizing its operators and re-ordering a polynomial within a subterm. So, for example, the functions within coefficients could be of arbitrary complexity.

5.3.1.2 Perspectives

As it stands, this is not a very informed way of preparing an equation for parsing analysis. The next stage is to choose a *perspective*. By treating certain sub-expressions as unexpandable units we can adopt different *perspectives* on the equation. Each *perspective* is

a rewriting of the equation to coincide with the syntax of one of an ensemble of normal forms. If subsequent parsing fails to find a *complete* and *consistent* interpretation in one *perspective* we may shift to another by expanding these coefficients.

For example, the weakly normalised differential equation

$$x^2*d^2y/dx^2 + 3*x^{-1}*(dy/dx)^3 + -1*(-1*x^2+1)*y = 0$$

can assume the following *perspectives*, in the order shown

$$x^2*d^2y/dx^2 + 3*x^{-1}*(dy/dx)^3 + -1*(-1*x^2+1)*y = 0 \quad (1)$$

$$x^2*d^2y/dx^2 + 3*x^{-1}*(dy/dx)^3 + x^2*y + -1*y = 0 \quad (2)$$

$$x^2*d^2y/dx^2 + 3*x^{-1}*(dy/dx)*(dy/dx)*(dy/dx) + -1*(-1*x^2+1)*y = 0 \quad (3)$$

$$x*x*d^2y/dx^2 + 3*x^{-1}*(dy/dx)^3 + -1*(-1*x*x+1)*y = 0 \quad (4)$$

$$x*x*d^2y/dx^2 + 3*x^{-1}*(dy/dx)^3 + x*x*y + -1*y = 0 \quad (5)$$

$$x^2*d^2y/dx^2 + 3*x^{-1}*(dy/dx)*(dy/dx)*(dy/dx) + x^2*y + -1*y = 0 \quad (6)$$

$$x*x*d^2y/dx^2 + 3*x^{-1}*(dy/dx)*(dy/dx)*(dy/dx) + -1*(-1*x^2+1)*y = 0 \quad (7)$$

$$x*x*d^2y/dx^2 + 3*x^{-1}*(dy/dx)*(dy/dx)*(dy/dx) + x*x*y + -1*y = 0 \quad (8)$$

Each perspective encodes progressively less parsimonious heuristic rewrites. The idea is to maintain as little elaboration as possible. There are two reasons for adopting this strategy. First, empirical observation suggests that the form of lumped coefficients can often yield useful clues as to the identity of certain function symbols. Second, treating coefficients as units keeps the complexity of the possible interaction pathways relatively low.

The enumeration order of perspectives is driven by the presence or absence of the following characteristics:

- whether * is distributed over + in the coefficients
- whether powers in the coefficients are expanded
- whether powers of derivatives are expanded

In general if we introduced n characteristics, each of which could be active or inactive, there would be 2^n possible *perspectives*. Therefore with 3 characteristics there are 8 *perspectives*. A possible improvement would be to invent more sophisticated "characteristics" (*e.g.* which considered simultaneous conjunctions of attributes) so that there would still be no-more than 2 or 3 dimensions in total but give rise to a more heuristically powerful set of *perspectives*.

Table 5-1 summarizes the setting of these characteristics for each of the perspectives (1) through (8).

	distribute * over +	expand powers in coefficients	expand powers of derivatives
Perspective(1)	×	×	×
Perspective(2)	✓	×	×
Perspective(3)	×	×	✓
Perspective(4)	×	✓	×
Perspective(5)	✓	✓	×
Perspective(6)	✓	×	✓
Perspective(7)	×	✓	✓
Perspective(8)	✓	✓	✓

Table 5-1. Features of Perspectives (1) to (8)

At the two extremes are *perspectives* (1) and (8). Perspective (1) is simply a weakly normalised differential equation which therefore retains coefficients (*e.g.* $(-1*x^2+1)$ above) as unexpanded units. Perspective (8) is a strong normal form for differential equations in the sense that all subterms are treated equally.

5.3.1.3 Mapping to a List

The next stage is to map the normalised equation into nested lists. Thus the expression

$$x^2*d^2y/dx^2 + 3*x^{-1}*(dy/dx)^3 + -1*(-1*x^2+1)*y = 0$$

becomes, in *perspective* (1), the nested list

```
[[x↑2,deriv(2,y,x)],[3,x↑-1,deriv(1,y,x)↑3],[-1,(-1*x↑2+1),y]].
```

The key points to note about this representation are as follows:

- It distinguishes between the operators in coefficients (*e.g.* those in $(-1*x^2+1)$) and those acting between terms; maintaining the former, suppressing the latter.
- The operators within nested lists are implicitly multiplicative and those between nested lists implicitly additive.
- Certain operators, *e.g.* the exponential, are retained.

5.3.2 Representation of the Interaction Graph

The nodes of the *interaction graph* are the terms in the expanded equation and the arcs are the inverse relationship proposed between the nodes it joins. By locating arcs judiciously we can home in on a labelling of the function symbols which satisfies all the proposed interactions simultaneously.

Two types of inverse relations (`+inv` and `*inv`) suffice for a wide range of differential equations. Their intended interpretations are that a term added to minus itself is zero, `+inv(X,-X,0)`, and a term multiplied by its own reciprocal is one, `*inv(X,1/X,1)`. We can express the tensions that exist between terms in a complicated equation by asserting that certain combinations are the additive or multiplicative inverses of others. A set of such inverse relations defines an *interaction graph*.

Thus an interaction graph is a collection of arcs, each of the form

```
arc(InverseType, LeftExpr, RightExpr, Byproduct)
```

where `InverseType` corresponds to either `+inv` or `*inv`, `LeftExpr` and `RightExpr` correspond to the two nodes at either ends of the arc and `Byproduct` represents the remainder after the interaction of `LeftExpr` and `RightExpr` under `InverseType`.

5.3.3 Representation of Context

As parsing proceeds various hypotheses are floated regarding the interpretation of function symbols. A maximally consistent subset of these define a *context* and are maintained as a list of *functor/analytic function* pairs. The *context* is used to guide the selection of rewrite rules at a later stage and also to assess termination state of the algorithm.

The construction of a *context* is rather like that of a parse tree in natural language processing. We can make the parsing process context sensitive by interpreting interaction arcs in the light of the current partial *context*. For example, we can use *context* information to test whether a particular rewrite rule for some subterm mentioned in an arc is currently valid. Ultimately, if **equation parsing** succeeds, we will obtain a consistent *context* which includes an interpretation for each function symbol in the functional form.

5.3.4 Representation of Function Clichés

5.3.4.1 Midget Equations

The properties of the function with respect to its derivatives (currently up to second order) are expressed *via midget equations*. *Midget equations* may contain the *function, its derivatives, powers of the independent variable* and *constants* but nothing else. For example, a *midget equation* for $f(x) = \sin(x)$ is "$d^2f/dx^2 + f = 0$". Similarly, one for $f(x) = \arccos(x)$ is

"$d^2f/dx^2 - x*(df/dx)^3 = 0$". It is straightforward to convert information in a *midget equation* to a *function cliché*.

5.3.4.2 Normal Form for Clichés

To aid intended matching processes it is necessary to define a normal form for clichés. Each cliché is asserted as a 5 argument Prolog fact.

```
cliche(InverseType, LeftExpr, RightExpr, Byproduct, Context)
```

The first argument asserts the inverse relationship that holds between the second and third arguments; the fourth argument is the term produced as a result of the interaction and the fifth one is an interpretation of the function symbols consistent with this behaviour.

The clichés relate three terms each of which may contain only unary negative, multiplicative or exponentiative operators. Thus they encode

```
LeftExpr + RightExpr = Byproduct
```

or

```
LeftExpr * RightExpr = Byproduct
```

The syntax of the `LeftExpr`, `RightExpr` and `Byproduct` of each cliché is as follows:

`LeftExpr` ::= $f(x)*[d^n y/dx^n]^p$

`RightExpr` ::= $g(x)*[d^{n-m} y/dx^{n-m}]^q \quad 0 < m \leq n$

`Byproduct` ::= $h(x)$

- $f(x)$, $g(x)$ and $h(x)$ may be a product of constants, powers of x and powers of y but nothing else. In particular there are no + operators within the subterms of a cliché.

- $f(x)$ and $g(x)$ may have a leading term "–1" but $h(x)$ may not

- numeric expressions are evaluated to reals (*e.g.* 3/2 becomes 1.5)
- repeated variables (including derivatives) are grouped into powers
- negative powers in `+inv` clichés are eliminated by multiplying through by the reciprocal
- `LeftExpr` contains the highest order derivative occurring in the *midget* equation from which the cliché is built
- `RightExpr` contains lower order derivatives, possibly just the function itself (*i.e.* zeroth order)
- `Byproduct` is always of lower derivative order than any term in `LeftExpr` or `RightExpr` and is in most cases a constant.

Examples derived from the above *midget equations* would be:

```
cliche(+inv, deriv(2,Y,X), Y, 0, Y=sin(X))
```

and

```
cliche(+inv, deriv(2,Y,X), -1*X*deriv(1,Y,X)↑3, 0, Y=arccos(X)).
```

More complex clichés include the following

```
cliche(+inv, deriv(2,Y,X), -1*Y, 0, Y=sinh(X)).

cliche(+inv, deriv(1,Y,X)↑2, -1*Y↑2, 1, Y=sinh(X)).

cliche(+inv, -1*deriv(2,Y,X)↑2, deriv(1,Y,X)↑2, 1, Y=sinh(X)).

cliche(+inv, -1*deriv(2,Y,X), 2*Y↑3, Y, Y=sec(X)).

cliche(+inv, -1*deriv(1,Y,X)↑2, Y↑4, Y↑2, Y=sec(X)).

cliche(+inv, Y*deriv(2,Y,X), -1*deriv(1,Y,X)↑2, Y↑4, Y=sec(X)).
```

The key point is that the `Byproduct` is always of lower complexity than the interaction it eliminates. As arcs in the interaction graph are represented as

```
arc(InverseType, LeftExpr, RightExpr, Byproduct)
```

it is straightforward to test whether a given arc has a known interpretation. However, in some cases an arc will contain a function which is not allowed in the normal form for clichés. To attempt to decode the arc it is then necessary to first rewrite the function to a derivative and either check or assert the *context* under which this has been done.

5.3.4.3 Substitution in Context

"Substitutions-in-context" are functional (as opposed to algebraic) rewrite rules which define equivalences between a function and a derivative in a *context*. Syntactically they are

```
sub-in-ctxt(Ctxt, Lhs, Rhs)
```

and are uni-directional `Lhs => Rhs`. For example, given the cliché

```
cliche(+inv, deriv(2,Y,X), deriv(1,Y,X)↑2, 0, Y=log(e,X)),
```

a substitution-in-context rewrite rule would be

```
sub-in-ctxt(Y=log(e,X), exp(-1*Y), deriv(1,Y,X)).
```

Substitutions-in-context permit proposed arcs to be rewritten to the normal form for clichés.

5.4 Equation Parsing

5.4.1 Conjecturing the Functional Form

We saw in the last chapter that, given a qualitative behaviour, **segment calculus** could be used to conjecture a compound form for it. The **equation parsing** technique, however, would be applicable regardless of how the conjectured compound form arose. For example it could be user supplied or generated on the basis of an analogical similarity to a known equation (there are actually theorems concerning just such analogical similarities – see, for example, [Kreider *et al* 80, pp231-240]). Consequently, the class of equations to which **equation parsing** may be applied is much wider than that of **analytic abduction** (which is currently restricted to autonomous equations due to the limitations of `QSIM`).

5.4.2 Algorithm

Algorithm 5-1. Parse Equation

Input

- a *differential equation* and
- a conjectured *functional form* for a solution

Method

- substitute the *functional form* into the equation and evaluate its derivatives
- choose a *perspective* on the equation

(*) • place subterms on an *agenda*

- initialise the *interaction graph* to be empty
- initialise the *context* to be empty

while *agenda* is non-empty **do**

(‡) • pick a pair of subterms from the *agenda*

(†) • propose an appropriate inverse relation (`+inv` or `*inv`)

- decode the interaction
- if a consistent interpretation is found,

- extend the *interaction graph* by installing an arc between the two subterms
- add any new interpretations of function symbols to the *context*
- remove the arc from the *agenda* and
- add the byproduct of the interaction to the *agenda*
- recurse from (‡)

- if no interpretation is found,
  - pick another pair of subterms and recurse from (†)

- if no consistent decoding is found, cache the partial accounting, choose a new *perspective* and recurse from (*)

5.4.3 Substitution into Differential Equation

Once a functional form has been proposed we can apply the normal rules of calculus to evaluate the form of its derivatives and substitute them into the differential equation. The resulting equation will usually contain more terms than the original but will also have a richer structure to it.

We next choose a *perspective* on the terms in the equation and proceed to build an *interaction graph*. At the moment *perspectives* are enumerated blindly in a pre-determined order. Future extensions could change this to take account of the modes of successes and failures of previous parsing attempts.

5.4.4 Building the Interaction Graph & Context

The *interaction graph* partitions the terms in the expanded equation with appropriate inverse relations. As parsing proceeds we attempt to decode each inverse relation, incrementally building a partial, maximally consistent labelling of function symbols with known analytic functions (*i.e.* a *context*). When no further subterms remain to be labelled we will have found a complete consistent labelling and the parsing terminates.

5.4.4.1 Choosing the Type of Inverse Relation

It is easy to decide which inverse relation to assert between two subterms by identifying the dominant functor between them. For example the dominant functor in "$X_1*X_2*X_3 + X_4*X_5 - X_6$" is "+". This is immediate from the nested list representation, [[X_1,X_2,X_3], [X_4,X_5], [–1,X_6]], where the operator between sub-lists is "+" and that between elements of sub-lists is "*". At any point the inverse relation asserted should be the inverse of the dominant functor.

5.4.4.2 Heuristics for Installing Interaction Arcs

It is not feasible, except for equations containing but a handful of terms, to exhaustively test all possible combinations of sub-terms as possible inverses of others. Some heuristic guidance is required.

Elimination Heuristic:

> Prefer arcs between sub-terms in which there are common factors which will cancel under the proposed interaction. For example, if an arc would encode $X_1*Y + X_2*Y = 0$ install it and examine $X_1 + X_2 = 0$.

Separability Heuristic:

> In many cases the functions comprising the functional form are linearly independent. This suggests we should prefer arcs which separate their dependencies. For example, if X's denote subterms containing f and Y's denote those containing g's then given an arc which encodes $X_1*Y_1+X_2*Y_2 = 0$ separability suggests looking at $X_1=-X_2$ and $Y_1=Y_2$ and *vice versa*.

Fragment Interaction Heuristic:

> If a subterm of one node bears a known inverse relation to a subterm of another then install an appropriate arc between them. This essentially encodes an opportunistic bottom up strategy.

Context Consistent Heuristics:

> If the analysis so far has led to an hypothesis as to the identity of some function, then seek subsequent cliché decodings which would preserve context consistency.

5.4.4.3 Decoding the Arcs in an Interaction Graph

In parsing equations, an arc may not express an interaction in a syntactically identical way to that in which it is stored as a function cliché. We therefore rewrite the arcs into the normal form for clichés so that they may be tested for equivalence. Each rewrite rule has various preconditions which test its applicability and post-conditions which modify the *context* if necessary. The preconditions test for the *types* of subterms involved in an arc, their *contained operators*, their *leading signs* and their *derivative orders*. This meta-information builds up a picture of features in the arc which need to be changed to bring it into the normal form of a cliché.

For example, suppose we knew the normal form versions of clichés pertaining to arc-cosine and secant:

```
cliche(+inv, deriv(2,Y,X), -1*X*deriv(1,Y,X)↑3, 0, Y=arccos(X))

cliche(+inv, Y*deriv(2,Y,X), -1*deriv(1,Y,X)↑2, Y↑4, Y=sec(X))
```

then the following operations are permissible:

Term Re-arrangement

This is achieved with rewrite rules encoding *algebraic* operations which place an arc in the same normal form as a cliché. This can involve eliminating negative powers, removing –1 from the `Byproduct`, evaluating arithmetic sub-expressions or permuting the `LeftExpr`, `RightExpr` and `Byproduct` to agree with the derivative ordering. For example, given the arcs

```
arc(+inv, (X↑-1)*deriv(2,Y,X), -1*deriv(1,Y,X)↑3, 0)
```

and

```
arc(+inv, Y*deriv(2,Y,X), (Y↑4 + -1*deriv(1,Y,X)↑2), 0)
```

the first would be normalised on eliminating the negative power by multiplying through by its reciprocal. The second would be normalised by partitioning the `RightExpr` and moving the `Y↑4` to the `Byproduct` position.

Term Substitution in Context

The second rewrite method encodes *functional* rewrites rather than purely *algebraic* ones. The precondition for the method is that the arc should contain terms whose "types" are incompatible with the normal form for a cliché (so-called *nasties*). If the *context* is already established (*i.e.* instantiated) we look for a substitution in context which would rewrite the *nasties* as derivatives in a manner consistent with the current context. Otherwise, if the context is currently free, any legitimate rewrite of the *nasties* is retrieved and the *context* is extended with the labelling of the function symbol required to justify this rewriting. This second procedure therefore modifies the *context* and returns the result in as a post-condition on the rule.

As the *context* changes some rewrites cease to be valid and others become possible. For example, suppose we were confronted with the arc

```
arc(+inv, deriv(2,Y,X), -1*X*(-1*X↑2 + 1)↑1.5, 0)
```

In the *context* {y=arccos(x)}, -1*(-1*x↑2 + 1)↑1.5 can be rewritten as deriv(1,y,x)↑-3.

This is accomplished by matching -1*(-1*X↑2 + 1)↑1.5 to the left hand sides of sub-in-ctxt rules. Although an identical expression cannot be found the matcher recognises (-1*X↑2 + 1)↑-0.5 can be scaled to the sought form. Invoking the rule

```
sub-in-ctxt(Y=arccos(X), (-1*X↑2 + 1)↑-0.5, -1*deriv(1,Y,X)
```

it is possible to rewrite -1*(-1*X↑2 + 1)↑1.5 to -1*(-1*deriv(1,Y,X))↑-3 which simplifies to deriv(1,y,x)↑-3.

If the context had not been instantiated the post-condition on executing the rule would have been that Y=arccos(x) where Y is a variable awaiting instantiation through future unifications.

Term Exponentiation

Recognition of a power of the presented arc is sometimes possible. The precondition is that the LeftExpr and RightExpr contain at least a subset of the variables occurring in some cliché, the Byproduct is zero and the *context*, if instantiated, is consistent. For example, given the following arc

```
arc(+inv, deriv(2,Y,X)↑2, -1*(X↑2)*(-1*X↑2 + 1)↑-3, 0)
```

and an attempted match to the cliché

```
cliche(+inv, deriv(2,Y,X), -1*X*deriv(1,Y,X)↑3, 0, Y=arccos(X))
```

this satisfies the precondition because of the presence of deriv(2,Y,X) and X. Such an arc is rewritten to an equality and the powers adjusted to agree with that of the highest derivative power in the matching cliché. This results in writing the original arc to either

```
arc(+inv, deriv(2,Y,X), -1*X*(-1*X↑2 + 1)↑-1.5, 0)
```

or

```
arc(+inv, deriv(2,Y,X), X*(-1*X↑2 + 1)↑-1.5, 0)
```

The ambiguity arises from taking the square root. The first arc is decoded with a substitution in context as described above. The second is entirely analogous but if the context, {`y=arccos(x)`} were carried over from the first part it would fail to be consistently decoded (because the second arc reduces to an `arcsin` cliché).

Extraneous Symbols which Cancel

Some interactions contain symbols which can be factored from the inverse relation. For example, the following arc

```
arc(+inv, N*deriv(2,Y,X), X*(deriv(1,Y,X)↑3)*N, 0)
```

contains the redundant symbol `N`. The precondition is that the same symbol should occur in `LeftExpr` and `RightExpr` with the same power if the inverse type is `+inv` and with reciprocal powers if it is `*inv`.

Note that certain `+inv` and `*inv` relations are interdependent as

```
arc(+inv, X, Y, 0) ≡ arc(*inv, X, -1*Y↑-1, 1)
```

but others are not so *e.g.* `arc(*inv, X, -1*Y↑-1, 0)` has no `+inv` form.

This means that we need only encode most *midget equations* as `+inv` relations. Moreover, equality (`=`) and being zero (`0`) are expressed in terms of `+inv` relations too:

```
X=Y ≡ arc(+inv, X, -Y, 0)
```

and

```
X=0 ≡arc(+inv, X, 0, 0).
```

5.4.5 Success Criteria & Termination

The **equation parsing** algorithm terminates with assured success when the *agenda* is empty and with apparent failure otherwise. We say "apparent failure" because an intriguing observation is that it is sometimes possible to construct the solution from only a subset of the *interaction graph*; the extra interaction(s) being merely confirmatory. In such cases it is entirely possible that the partial accounting is correct but the system does not have the requisite cliché knowledge to decode an interaction pair left on the *agenda*. This suggests two criteria we could use to assess the correctness of our interpretations of the *interaction graph*

- the *context* is consistent and complete with every *agenda* item being discharged
- the *context* is consistent and complete but there remain terms on the *agenda* which cannot be shown to be inconsistent with the *context*.

In both cases the *context* needs to contain a consistent labelling for every function symbol. To test the partial accounting for success, it would be necessary to substitute it into the original equation and try to simplify the equation to a tautology. However, it would be expensive to test every partial accounting in this way and time consuming to do so as parsing proceeds. Therefore a pragmatic solution is to cache only those partial accountings having a single unrecognised interaction and postpone testing them until the search for a complete parse, in a given *perspective*, has been exhausted. This provides **equation parsing** with a primitive learning capability.

5.5 Examples

5.5.1 Example 1

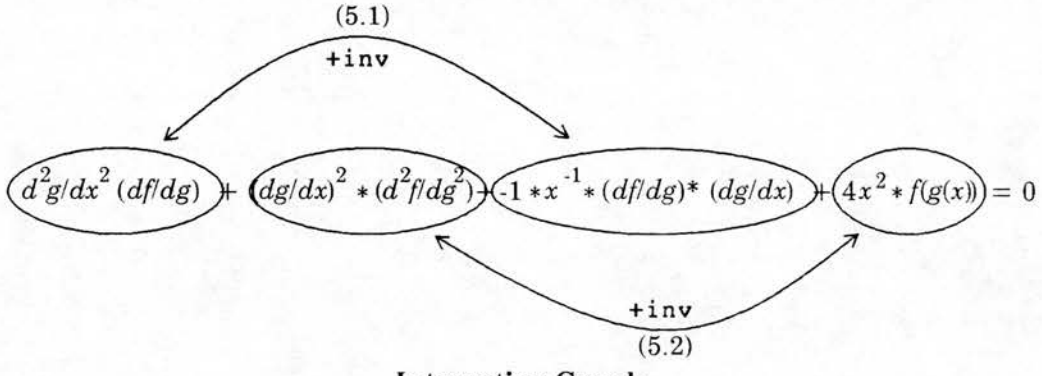
$$d^2y/dx^2 - (1/x)\, dy/dx + 4x^2y = 0$$

assumed form: $y = f(g(x))$ (composition)

Under this assumption the differential equation expands to a more complicated form whose individual termwise interactions are presumed simpler than those in the original equation.

$$d^2g/dx^2*(df/dg) + (dg/dx)^2*(d^2f/dg^2) + -1*x^{-1}(df/dg)*(dg/dx) + 4*x^2*f(g(x)) = 0$$

By postulating the following interaction graph it is possible to solve the equation.



Interaction Graph

Arc (5.1) is installed under the elimination heuristic. Arc (5.2) is then forced as there are only two subterms remaining. This process creates an *agenda* of arcs awaiting interpretation. As each arc is decoded the context in which it was recognised is passed on. This can be useful for finding *substitutions in context* which permit given arcs to be rewritten to alternate forms depending on the current labelling of function symbols.

Agenda

```
arc(+inv, deriv(2,g,x)*deriv(1,f,g),
          (-1*x↑(-1))*deriv(1,f,g)*deriv(1,g,x), 0)        (5.1)
arc(+inv, (deriv(1,g,x)↑2)*deriv(2,f,g), 4*(x↑2)*f, 0)     (5.2)
```

Under the separability heuristic, arc (5.2) is replaced by a pair of arcs which modify the structure of the original interaction graph. The new arcs are:

```
arc(+inv, (deriv(1,g,x)↑2), -1*4*(x↑2), 0) ∧
arc(+inv, deriv(2,f,g), f, 0)
```

Hence the new *agenda* is

Agenda

```
arc(+inv, (deriv(1,g,x)↑2), -1*4*(x↑2), 0)                    (5.3)
arc(+inv, deriv(2,f,g), f, 0)                                  (5.4)
arc(+inv, deriv(2,g,x)*deriv(1,f,g),
          (-1*x↑(-1))*deriv(1,f,g)*deriv(1,g,x), 0)            (5.1)
```

The first item on the *agenda* matches no known function cliché directly but can be decoded by considering its square roots. The relevant function cliché is

```
cliche(+inv, deriv(1,Y,X), -1*N*X↑(N-1), 0, Y=X↑N)
```

and the interpretation of arc (5.1) confirms this labelling. Arc (5.4) is decoded by the two clichés

```
cliche(+inv, deriv(2,Y,X), Y, 0, Y=sin(X))
cliche(+inv, deriv(2,Y,X), Y, 0, Y=cos(X))
```

So the final solution is $y = \sin(x^2) \vee y = \cos(x^2)$. By using partial knowledge of the structure of the solution (the *functional form*) it is possible to partition the terms of the expanded equation such that a complete and consistent *interaction graph* can be found.

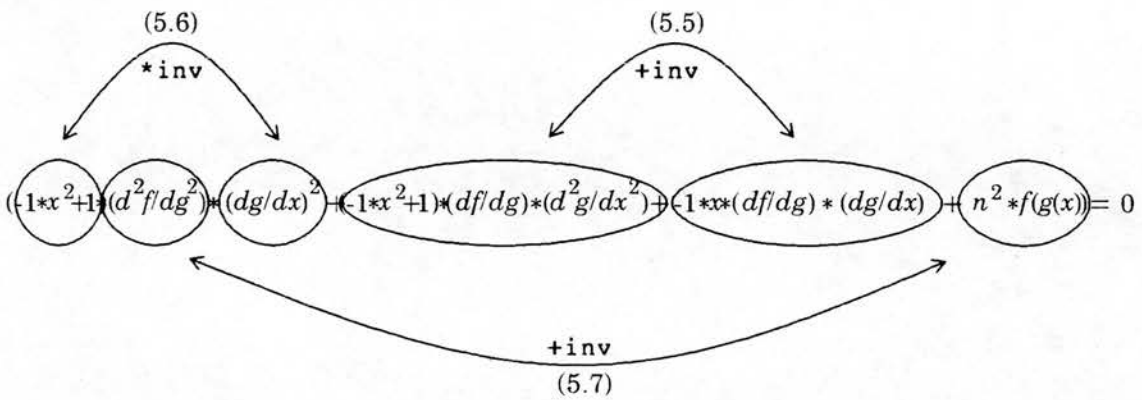
5.5.2 Example 2

$$(1 - x^2)d^2y/dx^2 - x*dy/dx + n^2*y = 0$$

assumed form: $y = f(g(x))$ (composition)

The input equation is placed in perspective (1) and the functional form substituted for y. This causes the equation to expand into the terms shown below.

$$(-1*x^2+1)*(d^2f/dg^2)*(dg/dx)^2+(-1*x^2+1)*df/dg*d^2g/dx^2+-1*x*df/dg*dg/dx+n^2*f(g(x))=0$$



Interaction Graph

Arc (5.5) is installed under the elimination heuristic, arc (5.7) is installed under the fragment interaction heuristic and arc (5.6) is forced as there are then only two terms remaining. Notice that this interaction graph contains a multiplicative inverse arc in addition to the usual additive inverse arcs. Hence, the agenda becomes:

Agenda

```
arc(+inv, (-1*x↑2+1)*deriv(f,1,g)*deriv(2,g,x),
                    -1*x*deriv(f,1,g)*deriv(1,g,x), 0)        (5.5)
arc(*inv, (-1*x↑2+1), deriv(1,g,x)↑2, 1)                      (5.6)
arc(+inv, deriv(2,f,g), (n↑2)*f 0)                            (5.7)
```

Arc (5.5) is decoded by deleting `deriv(f,1,g)` under the extraneous symbols heuristic and finding a substitution in context for `(-1*x↑2+1)`. As the system knows

```
sub-in-ctxt(Y=arccos(X), (-1*X↑2 + 1)↑-0.5, -1*deriv(1,Y,X))
```

it is easy to determine that `(-1*x↑2+1)` can be written as `deriv(1,Y,x)↑-2` in the context `{Y=arccos(x)}` for some uninstantiated variable `Y`. This is an example where substitution in context is used to assert a context as a post-condition on the rewrite rule. This reduces the arc to

```
arc(+inv, (deriv(1,Y,x)↑-2)*deriv(2,g,x), -1*x*deriv(1,g,x), 0)
```

The system normalises this by multiplying through by the reciprocal of the negative power.

```
arc(+inv, deriv(2,g,x), -1*x*(deriv(1,Y,x)↑2)*deriv(1,g,x), 0)
```

and makes the exploratory unification of `deriv(1,Y,x)` with `deriv(1,g,x)` as `g` is the only direct function of `x` in this trial function. Hence, the arc reduces to

```
arc(+inv, deriv(2,g,x), -1*x*deriv(1,g,x)↑3, 0)
```

which is easily decoded from the cliché

```
cliche(+inv, deriv(2,Y,X), -1*X*deriv(1,Y,X)↑3, 0, Y=arccos(X)).
```

Arc (5.6) is decoded using the cliché

```
cliche(+inv, deriv(1,Y,X), (-1*X↑2+1)↑-0.5, 0, Y=arccos(X))
```

Arc (5.7) is decoded using

```
cliche(+inv, deriv(2,Y,X), N↑2*Y, 0, Y=sin(N*X))
cliche(+inv, deriv(2,Y,X), N↑2*Y, 0, Y=cos(N*X))
```

In this case a *inv relation was coerced to a +inv form before being interpreted.

Hence the complete solution is

$$y = \cos(n*\arccos(x)) \ \vee \ y = \sin(n*\arccos(x))$$

5.5.3 Example 3

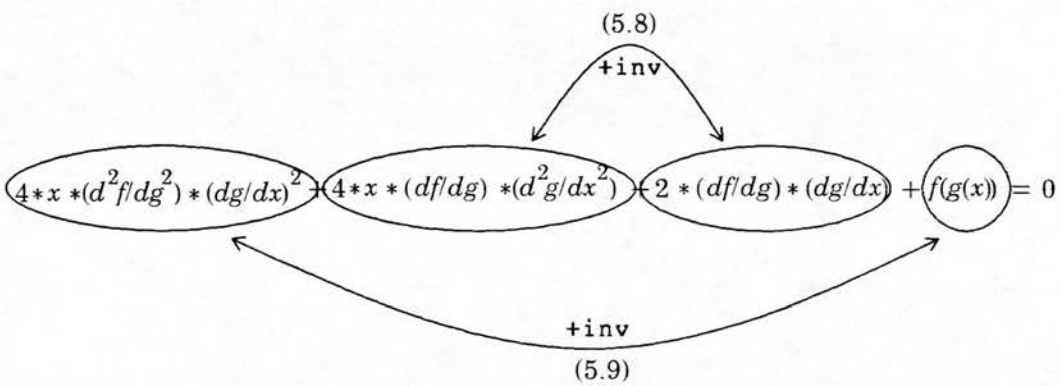
The following example may also be solved using **closed form approximation** discussed in *Chapter 7*. It is interesting to compare the two approaches. The equation is:

$$4*x*d^2y/dx^2 + 2*dy/dx + y = 0$$

assumed form: $y = f(g(x))$ (composition)

$$4*x*d^2f/dg^2*(dg/dx)^2 + 4*x*df/dg*d^2g/dx^2 + 2*df/dg*dg/dx + f(g(x)) = 0$$

which, under heuristic guidance, suggests the following *interaction graph*:



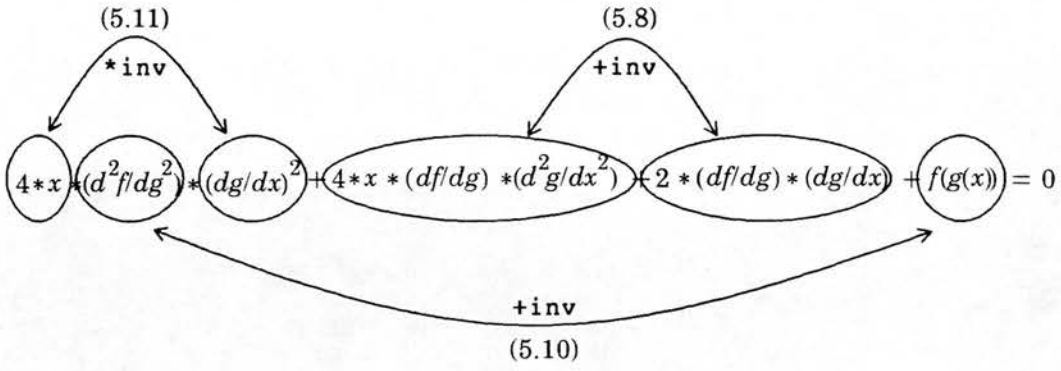
Interaction Graph

The central arc is added because of the **elimination heuristic** (*df/dg* occurs in both central terms) and the lower arc is forced. Hence the initial *agenda* is:

Agenda

```
arc(+inv, 4*x*deriv(1,f,g)*deriv(2,g,x),
                    2*deriv(1,f,g)*deriv(1,g,x), 0)          (5.8)
arc(+inv, 4*x*deriv(2,f,g)*(deriv(1,g,x)↑2), f, 0)           (5.9)
```

Under the **fragment interaction heuristic** the bottom arc is restructured to make the interaction between d^2f/dg^2 and f explicit and a `*inv` arc is installed to account for the remainder. This modifies the *interaction graph* to the form shown:



Interaction Graph

and hence the *agenda* becomes:

Agenda

```
arc(+inv, deriv(2,f,g), f, 0)                                    (5.10)
arc(*inv, 4*x, (deriv(1,g,x)↑2), 1)                              (5.11)
arc(+inv, 4*x*deriv(1,f,g)*deriv(2,g,x),
                    2*deriv(1,f,g)*deriv(1,g,x), 0)              (5.8)
```

Arc (5.10) is easily recognised from

```
cliche(+inv, deriv(2,Y,X), Y, 0, Y=sin(X))
cliche(+inv, deriv(2,Y,X), Y, 0, Y=cos(X))
```

as before.

Arc (5.11) encodes the relation `(4*x)*(deriv(1,g,x)↑2) = 1`. By taking the square root and mapping the multiplicative inverse to an additive inverse form, (5.10) can be decoded using

```
cliche(+inv, deriv(1,Y,X), A*X↑B, 0, Y=M*X↑N)
```

where `M` and `N` are evaluated from `M=A/(B+1)` and `N=B+1`. This yields `g = ±x↑(0.5)`.

Finally, (5.8) confirms the interpretation of (5.11) using the function cliché

```
cliche(+inv, A*deriv(2,Y,X), deriv(1,Y,X), 0, Y=X↑N)
```

where `N=1-(1/A)`. Notice however, that it is unnecessary for the system to know of this cliché because, at the point it attempts to decode arc (5.8), the hypothesis `g = ±x↑(0.5)` is already known. Consequently, arc (5.8) can be verified simply by direct evaluation. In many circumstances direct evaluation is the only way to verify an arc. However, having done so the system can record the interaction as a new cliché. This amounts to a primitive learning capability.

Collecting the interpretations together, the final solution is

$$y = \sin(x^{1/2}) \vee \cos(x^{1/2})$$

This problem is interesting in that it may also be solved by **closed form approximation** (another of the techniques we have developed) although the manner of solution is quite different. The problem is re-worked using **closed form approximation** in the next chapter.

5.5.4 Example 4

This example, posed in a standard textbook [Boas 66 p575], cannot be solved using **closed form approximation** but can be solved using **equation parsing**. The equation is:

$$x^4 * d^2y/dx^2 + y = 0$$

and the **assumed form**: $y = f(x)*g(h(x))$ (product containing composition)

Substitute into differential equation:

$$x^4*g*d^2f/dx^2 + 2*x^4*df/dx*dh/dx*dg/dh + x^4*f*d^2h/dx^2*dg/dh + x^4*f*d^2g/dh^2*(dh/dx)^2 + f*g$$

$$= 0$$

(1) + (2) + (3) + (4) + (5)

The next step is to set up an initial interaction graph. A `+inv` interaction arc could be installed between various pairs of terms (labelled (1) through (5)) in compliance with the **elimination heuristic**. Each pair is inspected to determine its largest common subterm. The results are shown in Table 5-2 below.

Interaction Pair	Common Subterm
(1)&(2)	x^4
(1)&(3)	x^4
(1)&(4)	x^4
(1)&(5)	g
(2)&(3)	$x^4 * dg/dh$
(2)&(4)	$x^4 * dh/dx$
(2)&(5)	$\varnothing$
(3)&(4)	$x^4 * f$
(3)&(5)	f
(4)&(5)	f

Table 5-2. Subterms Common to Each Interaction Pair

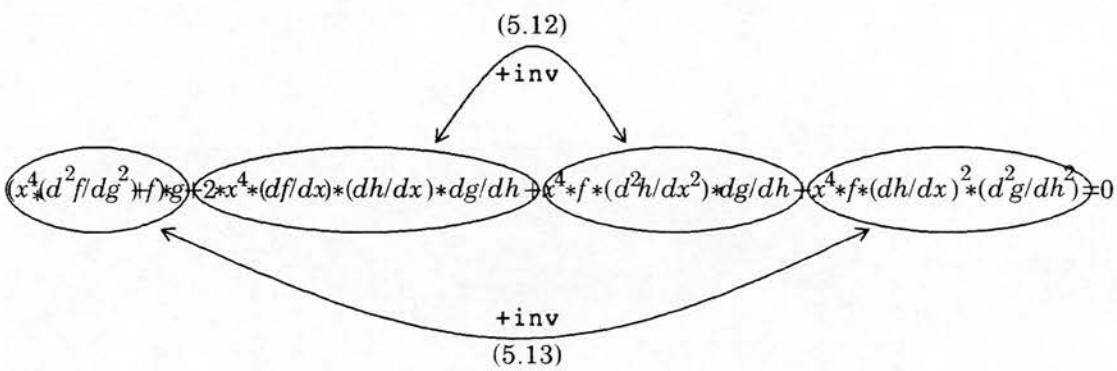
The system decides that the pairing between (2)&(3) is the most promising as it eliminates the most complex common subterm. Three terms then remain. As each interaction is restricted to be between two subterms it is necessary to merge two of them together.

The system analyzes the subterms to determine the ones most likely to be merged successfully. Grouping (1) with (4) or (4) with (5) would require combining terms in f, g, and h. However, grouping (1) with (5) would only require combining terms in f and g. Hence the latter possibility is preferred. In forming the merger the system factors out common subterms to try to write the combination such that the dominant functor is *. This is because

the heuristics for decoding interaction arcs presume the subterms are in multiplicative form. Consequently, the equation is written in the form

$$(x^4*d^2f/dx^2+f)*g+2*x^4*df/dx*dh/dx*dg/dh+x^4*f*d^2h/dx^2*dg/dh+x^4*f*d^2g/dh^2*(dh/dx)^2 = 0$$

Under the **fragment interaction heuristic** we can foresee a relation between g and d^2g/dh^2. Hence the initial interaction graph is:



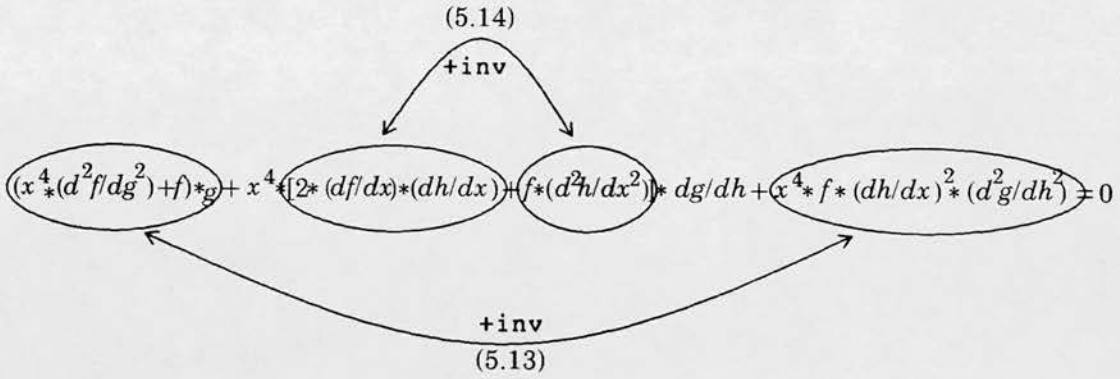
Interaction Graph

and the *agenda* is initialized with two additive inverse relations each node of which is a fairly complex subterm.

Agenda

```
arc(+inv, 2*(x↑4)*deriv(1,f,x)*deriv(1,h,x)*deriv(1,g,h),
          (x↑4)*f*deriv(2,h,x)*deriv(1,g,h), 0)                (5.12)

arc(+inv, [(x↑4)*deriv(2,f,x) + f]*g,
          (x↑4)*f*deriv(2,g,h)*(deriv(1,h,x)↑2), 0)            (5.13)
```

Under the **elimination** heuristic (5.12) simplifies by factoring out the `(x↑4)*deriv(1,g,h)`,



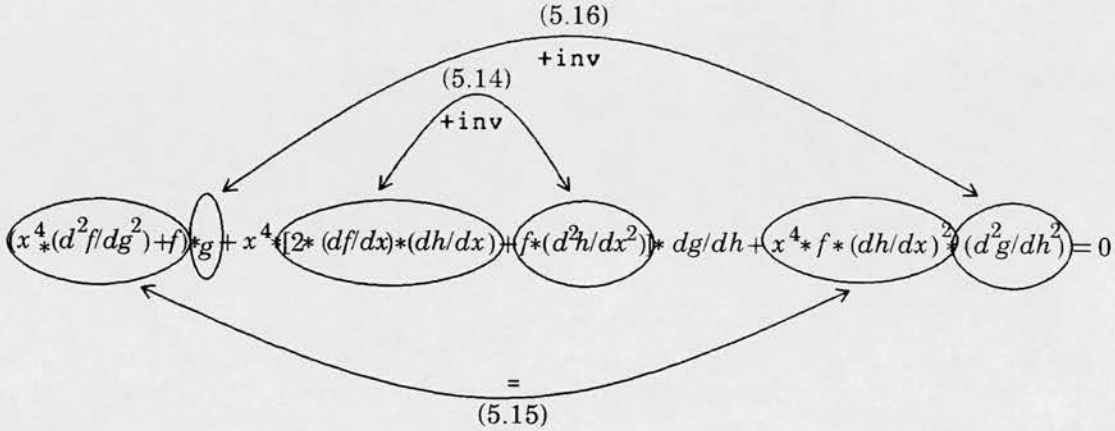
Interaction Graph

making the new *agenda*

Agenda

```
arc(+inv, 2*deriv(1,f,x)*deriv(1,h,x), f*deriv(2,h,x), 0)        (5.14)

arc(+inv, [(x↑4)*deriv(2,f,x) + f]*g,
          (x↑4)*f*deriv(2,g,h)*(deriv(1,h,x)↑2), 0)              (5.13)
```

Applying the **separability** heuristic, (5.13) can be partitioned into two terms.†



Interaction Graph

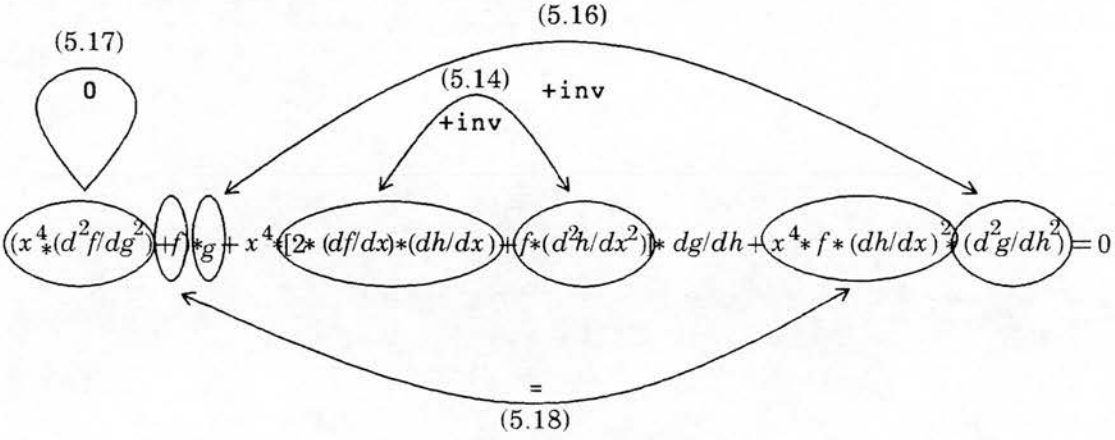
Agenda

```
arc(+inv, [(x↑4)*deriv(2,f,x) + f],
          -1*(x↑4)*f*(deriv(1,h,x)↑2) , 0)                       (5.15)

arc(+inv, g, deriv(2,g,h), 0)                                    (5.16)

arc(+inv, 2*deriv(1,f,x)*deriv(1,h,x), f*deriv(2,h,x), 0)        (5.14)
```

† Recall (page 166) that the relations `X=Y` and `X=0` can be expressed in terms of the additive inverse relations `+inv(X,-Y,0)` and `+inv(X,0,0)` respectively. For clarity, arcs encoding these relations are labelled "=" and "0" in an interaction graph.

To continue the analysis it is necessary to merge the two terms in the `LeftExpr` of arc (5.15). The simplest way of doing this is to assume one member of the pair is zero. Clearly, $f \neq 0$ for a non-trivial solution. Hence the system hypothesises $d^2f/dx^2 = 0$ and examines the ramifications.

Under the **elimination heuristic** an arc between f and $x^4*f*(dh/dx)^2$ would be promising anyway as both terms contain f in a factorable position.



Interaction Graph

Hence the *agenda* becomes

Agenda

```
arc(+inv, (x↑4)*deriv(2,f,x), 0, 0)                          (5.17)
arc(+inv, f, -1*(x↑4)*f*(deriv(1,h,x)↑2), 0)                 (5.18)
arc(+inv, g, deriv(2,g,h), 0)                                (5.16)
arc(+inv, 2*deriv(1,f,x)*deriv(1,h,x), f*deriv(2,h,x), 0)    (5.14)
```

Arc (5.17) may be decoded using the cliché

```
cliche(+inv, deriv(2,Y,X), 0, 0, Y=C*X)
```

Arc (5.18) simplifies to `arc(+inv, 1, -1*(x↑4)*(deriv(1,h,x)↑2), 0)`. By re-arranging and taking the square root it is can be decoded using

```
cliche(+inv, deriv(1,Y,X), A*X↑B, 0, Y=M*X↑N)
```

where **M** and **N** are evaluated from **M=A/(B+1)** and **N=B+1.**

Arc (5.16) is directly decodable from the clichés

```
cliche(+inv, deriv(2,Y,X), Y, 0, Y=sin(X))
cliche(+inv, deriv(2,Y,X), Y, 0, Y=cos(X))
```

Finally arc (5.14) can be verified by direct evaluation using the context generated from interpreting the other arcs *i.e.* using the labelling $\{f=C*x$ (for some constant C), $h=\pm x^{-1}$, $g=\sin(h)$ or $\cos(h)\}$.

Hence the final solution is

$$y = x*(c_1*\cos(\pm 1/x) + c_2*\sin(\pm 1/x))$$

5.6 Parsing Issues

5.6.1 Graph Grammars

We have adopted a very simple representation of equations as our major concern in this chapter was to establish the *principle* underlying **equation parsing** without worrying too much about implementation details. However, a better formalism would be graph grammars [Ehrig 79, Lutz 86, Pfaltz & Rosenfeld 69, Rosenfeld & Milgram 72].

A graph grammar extends the concepts of string parsing (one dimensional structures) to graphs (multi-dimensional structures) by encoding the possible ways in which one graph may embed in another. Many parsing concepts carry over from the string case including chart parsing [Lutz 86]. Recasting **equation parsing** in a graph grammar formalism may be more elegant but it is not clear that it will be more efficient as graph grammar parsers are very computationally expensive. For our application future effort may be better expended on developing more powerful heuristics for choosing where to install interaction arcs.

5.6.2 Top Down versus Bottom Up

As in most natural language systems it should be possible to parse the equation top down or bottom up. In the top down mode we would proceed by proposing interaction arcs between subterms in the expanded equation and attempt to decode them in terms of known function clichés. Conversely, bottom up parsing would examine the known function clichés and attempt to match them to fragments of the expanded equation. We should allow for the possibility of installing arcs which match one member of the interaction pair as there could be a substitution in context which would map the destination node into the other member of the interaction arc even if it were not immediately recognized.

Bottom up parsing requires explicit *function clichés* (derived from *midget equations*). Hence, it would not be possible if we had chosen to interpret the arcs in the interaction graph using symbolic integration of the simple differential equations implied by the arcs.

It is possible that a mathematician uses both modes of parsing simultaneously (*i.e.* bi-directional parsing). This may prove to be optimal *e.g.* by using the data to suggest the placement of interaction arcs (bottom up) and the equation to suggest others (top down) with the equations implied by interaction arcs integrated symbolically. An example of bi-directional parsing may be found in [Steel & de Roeck 87].

5.6.3 Caching Results

As parsing proceeds we should cache the interpretations of interaction pairs as and when they are found (as in chart parsing or truth maintenance systems [de Kleer 86]) so that we can avoid re-partitioning the equation in the same way again and again each time looking for an interpretation in a different context. In other words, we should parse in multiple contexts simultaneously.

5.7 Limitations

If the differential operator is linear and the proposed functional form of the solution is a sum of two functions then nothing can be gained from **equation parsing**. Substituting the form $h(x) = f(x)+g(x)$ into a linear operator $L[h(x)]$ will only yield $L[f(x)]$ and $L[g(x)]$ which provides no finer structure and hence no new information with which to guide a parse.

Relying solely on clichéd knowledge to recognise (*i.e.* effectively integrate) the simpler differential equations implied by interaction arcs is a weak (but psychologically plausible) method. A more robust procedure could be built by using a symbolic integration package. However, we should still like to retain explicit clichés to permit a bottom up parsing strategy.

The manner in which perspectives are selected could also be improved perhaps by allowing fragments of coefficients to be matched to parts of function clichés. This might allow for a more a informed choice.

The current equation parser is only a prototype and we believe its performance could be considerably enhanced using truth maintenance, alternative heuristics for choosing perspectives and installing interaction arcs and more controlled movement between perspectives.

5.8 Conclusions

The parsing approach to differential equation solving provides a structured way of attempting to conjecture the closed form solution without solving the equation in the traditional sense.

Equation parsing reveals particular solutions of equations and is equally applicable to linear or nonlinear equations.

Parsing and decoding are distinct operations and the cliché recognition phase could be replaced by a proper symbolic equation solver which would make the procedure more robust. However, explicit clichéd knowledge is necessary if a bottom up or bi-directional approach is desired.

The notion of seeking a family of inverse relations which specify an alternate structure to an equation capturing the mechanics of the termwise interactions should be more widely applicable than in differential equation solving. If in other domains we can also find partial information about the structure of the solution then we might also be able to find a novel solution technique similar to that described in this chapter.

The progression through *perspectives* is an attempt to capture a more disciplined approach to enumerating the ways in which an equation can be perceived. However, it is really a question of principle rather than pragmatics. Ideally, we want to to catch the "correct" parse of an equation from perspective (1) so that no further perspectives are ever explored.

I have not yet implemented anything other than blind enumeration of perspectives and this is truly miserable when things go wrong. However, it might be possible to adopt a more informed choice of which perspective to move to next.

For the future, an interesting way to proceed would be to assemble a large set of differential equations which are known to have closed form solutions (it is trivial to generate these automatically!) and use machine learning to hypothesise new heuristics for choosing perspectives and installing interaction arcs.

Equation parsing is not an approximate technique as discussed in the other chapters of this thesis; if it succeeds it produces an exact, particular solution. It might seem tempting to make it the basis for an approximate method by weakening the requirement for *completeness* in the *interaction graph*. However, we would urge caution. **Equation parsing** has no concept of the qualitative properties of the solution it is attempting to find. A

symbolic solution which is formally "close" (in some sense) to satisfying the differential equation might be a poor choice with respect to the qualitative behaviour of the real solution.

Ironically, this highlights the power of **analytic abduction**. By focusing on qualitative behaviour **analytic abduction**'s approximations are genuinely useful, even if rather coarse. Without preserving qualitative fidelity approximations can be pathological.

Finally we should point out that both **equation parsing** and **analytic abduction** are limited in the equations they can handle. **Equation parsing** implicitly assumes the equation *has* a closed form solution. **Analytic abduction** can only accommodate autonomous equations because of the use of the `QSIM` algorithm. The next chapter introduces a final solution technique, **closed form approximation**, which goes some way towards ameliorating both problems.

Chapter 6

Solution in Series

6.1 Introduction

Analytic abduction, **Segment Calculus** and **Equation Parsing** all approach the problem of obtaining functional approximations to the behaviour of physical systems by reasoning with an abstraction of the original equation and mapping the results back to the mathematical level. In this chapter we begin our study of an alternative strategy based on solving the equation exactly and approximating the exact solution.

This approach is similar to that employed by Sacks in his **QMR** system [Sacks 85a, 85b], although the intention there was to map the exact solution to a qualitative description rather than an approximate functional one. However, the technique of **back of the envelope reasoning**, introduced in *Chapter 8*, could be used to generate functional approximations, in a magnitude extreme, to **QMR** output. A much more interesting research question is what to do about the sorts of equations **QMR** cannot handle *i.e.* those whose solutions cannot be written in closed form? Indeed Sacks himself acknowledged this problem [Sacks 85a p139] but subsequently turned his attention to piecewise linear approximations of nonlinear equations [Sacks 87b, 88] rather than tackling singular equations directly.

In physical applications, however, singular equations are fairly common, usually being associated with spatially symmetric problems. In fact the ubiquitous orthogonal polynomials of mathematical physics, which provide the basis for describing a wealth of physical phenomena, arise as solutions of the class of equation we will be considering. So, although we might appear to be tackling only a small class of problem it is important to realize that it is a significant class.

The problem of approximation divides naturally into two phases. First the exact infinite series solution must be obtained and second, a closed form expression must be constructed which approximates it sufficiently well. This chapter concerns the first phase; the next chapter will address the second.

Our plan is a follows. We begin by reviewing the mathematical basis for solution in series and describe the types of equations to be considered. Then we show how a theorem from analysis can be used to circumvent most of the working usually presented in text book examples of the method. This makes it possible to devise a computationally efficient algorithm for obtaining infinite series solutions. However, the naive representation of such series is unwieldy and a novel representation is introduced.

It is useful to introduce some terminology and sketch the components of a program to automatically generate closed form approximations. First we shall call the infinite series which is the exact solution of the differential equation the **target** series. Second we call any series which is "close" to the target in the sense outlined above a **candidate** series. In general there will be many candidates, each a different distance from the target. The best candidate is the one whose distance to the target is least and we call this the **twin**.

To create candidate series we will need a set of **base** series which are the Maclaurin expansions of known closed form functions such as $\sin(x)$, $\exp(x)$ *etc* and an understanding of how mathematical operations such as multiplication or composition with a polynomial transform infinite series.

We make no claims as to the originality of the material in §6.2. The theory of series solutions has been developed over many years by scores of mathematicians. However, it is

important to include it in the body of the thesis as the subsequent **closed form approximation** technique could not be described without it.

6.2 Mathematical Basis for Solution in Series

It is not possible to write the solutions of all differential equations in closed form [Kreider *et al* 80] *i.e.* as a combination of the elementary mathematical functions we all learn at school. If the equation does not have a closed form solution then clearly **equation parsing** and **QMR** will not be capable of solving it. In such cases we must be content to solve the equation as an infinite series convergent in some interval [Kreider *et al* 80, Boas 66, Stephenson 78, Coddington 62]. However, we should like to arrange things such that if the differential equation does have a closed form solution the solution in series technique will yield precisely its power series expansion. The inspiration behind the technique is an existence theorem [Kreider *et al.* 80 pp243-244] which says that if the coefficients of a certain class of differential equations are analytic then so are its solutions. The fact that the solution is analytic is sufficient to guarantee that it can be written as a power series. Knowledge of the general form of the solution, together with the original equation, is then sufficient to determine the coefficients in it.

We consider second order equations (although the concepts readily generalise), outline the rationale behind solution in series and define necessary preconditions which test the applicability of the method.

6.2.1 Normal Equations and Power Series Solutions

Consider the equation

$$c_2(x)\, d^2y/dx^2 + c_1(x)\, dy/dx + c_0(x)\, y = 0$$

where c_2, c_1 and c_0 are analytic about $x_0 \in I$ (the intersection of the radii of convergence of the respective power series expansions of c_0, c_1, c_2). If $c_2(x) \neq 0$ for all $x \in$ I then we can rewrite the equation as

$$d^2y/dx^2 + q(x)\, dy/dx + r(x)\, y = 0$$

where $q(x)$ and $r(x)$ are both analytic at $x_0 \in I$ and hence can be expanded as power series about x_0

$$q(x) = \sum_{i=0}^{\infty} q_i(x-x_0)^i \quad \text{and} \quad r(x) = \sum_{i=0}^{\infty} r_i(x-x_0)^i$$

which are convergent within some radii of convergence R_q and R_r respectively. In this case, $x=x_0$ is said to be an ordinary point of the differential equation and we can always find a solution of the form

$$y = \sum_{i=0}^{\infty} a_i x^i \ \wedge \ a_0 \neq 0$$

convergent at least within $\min(R_q, R_r)$. By substituting the general form into the differential equation, evaluating derivatives and equating coefficients of corresponding powers of x to zero we arrive at the following *recurrence relation* [Kreider *et al.* 80 p255] which links later coefficients in the series to earlier ones.

$$a_{i+2} = -[\, 1/((i+1)(i+2))\,] \sum_{j=0}^{i} [(j+1)q_{k-j} + r_{k-j}\, a_j]$$

6.2.2 Non-normal Equations and Generalised Power Series Solutions

Many equations arising in practice are more troublesome in the sense that it will often be the case that $c_2(x_0) = 0$ for some $x_0 \in$ I. However, as the coefficients c_0, c_1, c_2 are analytic, each has a power series expansion about x_0. In particular

$$c_2(x) = \sum_{i=0}^{\infty} a_i(x-x_0)^i$$

But as $c_2(x_0) = 0$, the constant term in this series (*i.e.* the coefficient of $(x-x_0)^0$) must be zero so $c_2(x)$ can be rewritten as

$$c_2(x) = \sum_{i=m}^{\infty} a_i(x-x_0)^i$$

for some integer, $m \geq 1$ or, rewriting the right hand series in power series form

$$c_2(x) = (x-x_0)^m \sum_{i=m}^{\infty} a_i(x-x_0)^{i-m} = (x-x_0)^m\, c_3(x)$$

where $c_3(x)$ is analytic at x_0 and $c_3(x_0) \neq 0$. Consequently, we can always divide through by $c_3(x)$ and place a non-normal equation in the form

$$(x-x_0)^m\, d^2y/dx^2 + q_1(x)\, dy/dx + r_1(x)\, y = 0$$

where $q_1(x) = c_1(x)/c_3(x)$ and $r_1(x) = c_0(x)/c_3(x)$. Moreover, we can assume without loss of generality, that the singularity is at the origin (otherwise we simply transform to the new coordinate system $x'=x-x_0$). In keeping with almost every text book on the subject, we further restrict ourselves to

$$x^2\, d^2y/dx^2 + xq(x)\, dy/dx + r(x)\, y = 0$$

written as

$$d^2y/dx^2 + (q(x)/x)\, dy/dx + (r(x)/x^2)\, y = 0$$

which facilitates comparison with the ordinary point case. We now see that if $q(x)$ and $r(x)$ are analytic at $x=x_0$ (*i.e.* expandable in terms of convergent power series) we can always find at least one solution of this equation in the form. For convenience we refer to $q(x)/x$ and $r(x)/x^2$ having these properties as *pseudo-analytic* functions. In the vicinity of a regular singular point we can always find at least one solution of the differential equation of the form

$$y = \sum_{i=0}^{\infty} a_i x^{i+s} \quad \wedge \quad a_0 \neq 0$$

where s is not necessarily an integer and the series is convergent at least within the smaller of the radius of convergence of $q(x)$ or $r(x)$. Moreover, the structure of the solution may be summarised by the following theorem.

Theorem 6-1. Structure of the Recurrence Relations

[Kreider *et al* 80 pp582-593] states that for all regular singular equations the indicial equation has the form

$$s(s-1) + q(0)s + r(0) = 0$$

and the recurrence relations are

$$a_i = -1/(I(i+s)) \sum_{j=0}^{i-1} [(j+s)\, q_{i-j} + r_{i-j}]\, a_j, \quad i \geq 1$$

where $I(x) = x(x-1) + q(0)\, x + r(0)$, there being one recurrence relation for each value of the indicial index s.

As the indicial equation is quadratic there will in general be two solutions for the indicial index, s. If these are equal or differ by an integer, some a_i other than a_0 will usually become indeterminate and the infinite series will contain more than one undetermined coefficient. For these the method described will in general yield only one solution.

The solution in series technique we introduce can accommodate both ordinary and regular singular points.

6.2.3 Solution in Series Algorithm

The above results can be exploited to conceive a method for solving ordinary differential equations including those that do not have closed form solutions:

1) given an input differential equation, determine the *recurrence structure* of its solutions

2) given a *recurrence structure*, derive the associated infinite series

The traditional approach, presented in textbooks, is rather long-winded. First, the abstract form for the infinite series is substituted into the differential equation and the appropriate derivatives evaluated. This is permissible because the series solution is guaranteed to be analytic [Kreider *et al.* 80 pp243-244] and can therefore be written as a power series. It is possible to differentiate a power series term by term to generate another power series having the same radius of convergence [Kreider *et al.* 80 p663]. Then, the uniqueness property of series expansions [Kreider *et al.* 80 p664] guarantees that if the two sides of the equation are to balance, they must represent the same series. Hence, because of linear independence, the coefficients of corresponding powers of x must balance separately. In general this allows a system of equations, called *recurrence relations*, to be set up, which link later coefficients in the series to earlier ones. The recurrence relations are then interpreted to yield the associated infinite series solution.

Emulating these steps on a computer would be a formidable undertaking. Fortunately, the abstract expressions for the recurrence relations derived in [Kreider *et al* 80] can be exploited to yield a fast algorithm for mapping the input equation into a recurrence structure. It surprises me that I have never come across a single text book example which used these formulae directly, preferring instead to proceed from first principles. From a computational perspective the abstract recurrence formulae are precisely what is needed to make solution in series a tractable proposition and frees us to focus all our computational effort on finding a closed form approximation.

The second stage of mapping the recurrence relations into an infinite series is also of interest computationally. In the hand-cranked mathematical scheme there is no question about the representation of infinite series. However, from a our point of view, if we employed the standard representation we would incur large computational overheads in performing the kind of mathematical operations to be considered in the next chapter. Consequently, we developed a novel representation which will be described in §6.4.5.

However, we postpone further discussion until after a more detailed description of the steps in mapping the input differential equation into a recurrence structure.

6.3 Step (1): Input Equation to Recurrence Structure

The first step amounts to automating the solution in series method for solving differential equations. To do this efficiently we exploit Theorem 6-1 which stated that for all differential equations of the form

$$d^2y/dx^2 + (q(x)/x)\, dy/dx + (r(x)/x^2)\, y = 0$$

the indicial equation will always be

$$s(s-1) + q(0)s + r(0) = 0$$

and the recurrence relation given by

$$a_i = -[\, 1/(I(i+s))\,] \sum_{j=0}^{i-1} [(j+s)\, q_{i-j} + r_{i-j}]\, a_j, \quad i \geq 1$$

where $I(x) = x(x-1) + q(0)\, x + r(0)$. This neatly sidesteps the large amounts of algebraic manipulation present in textbook examples of solution in series.

Algorithm 6-1. Map Differential Equation to Recurrence Structure

1i)	put equation in normal form
1ii)	verify conditions for solution in series satisfied
1iii)	compute expansions for $q(x)$ and $r(x)$
1iv)	evaluate $q(0)$ and $r(0)$, set up and solve indicial equation
1v)	compute general form for recurrence relation
1vi)	simplify recurrence relation
1vii)	for each indicial index, identify corresponding selectors
1viii)	create recurrence structure

We place the equation in a normal form so that we can run a standard test on it to check that the method of solution in series is applicable [Boas 66]. This verification process consists of confirming that $q(x)$ and $r(x)$ can be written as power series. Their expansions could either be retrieved from stored examples in a library or calculated using the general formula for a Maclaurin expansion (only the first few terms will ever be needed). The indicial equation is easily solved using the techniques found in **PRESS**.

The recurrence relations arise by instantiating and simplifying Theorem 6-1. The recurrence relation simplification is guided by the general form we aim to write the recurrence relation in. Although it looks complicated, the recurrence structure usually collapses to something much simpler depending on the complexity of $q(x)$ and $r(x)$. For the systems we consider this will always be

$$a_i = \sigma * f(i;s) * a_{i-n}$$

which is a difference equation containing a leading sign $\sigma = +1$ or -1 and a strictly positive function of the subscript i, $f(i;s)$, parameterised by the indicial index s. The key features to recognise are that the spacing between indices is n and that the coefficients of the series will alternate in sign if $\sigma = -1$.

Precisely which coefficients are non-zero depends on which of the a_i are non-zero (for $0 < i < n$). In general a different set will be non-zero for each value of the indicial index s. If we define $k^{(s)}$ to be the set of subscripts of coefficients for an indicial index s such that $\forall k \in k^{(s)}$, $a_k \neq 0 \wedge k < n$ then the *selectors*, K, is then the collection of all such sets $\{k^{(s)}\}$.

Bringing these results together suggests the information required to specify an infinite series solution are the possible values for the *indicial indices*, a *recurrence relation* and a set of *selectors* (the subscripts of non-zero and indeterminate coefficients in the **target** series).

$$\langle \textit{recurrence relation, selectors, indicial indices} \rangle =$$

$$\langle b_i = \sigma * f(i;s) * b_{i-n},\ \{\{k^{s'}{}_1, k^{s'}{}_2, \ldots, k^{s'}{}_N\}, \{k^{s''}{}_1, k^{s''}{}_2, \ldots, k^{s''}{}_M\}\}, \{s', s''\} \rangle$$

We call such a tuple a *recurrence structure*.

6.4 Representation for Infinite Series

The next task is to derive the **target** series defined by the *recurrence structure*. To do this it is necessary to decide upon a representation for infinite series.

6.4.1 Why Representation is Important

Although mathematicians have used a formalism for representing infinite series for centuries, from a computational perspective, representation is critical and if we base our manipulations on the standard notations we incur large computational overheads. This motivated the development of a novel representation whose key feature is that we reason with the *properties* of the infinite series rather than the series itself. Whenever we need to exhibit the actual series it therefore has to be generated from this meta-representation. The advantage is that the meta-representation is more easily derived from the recurrence structure than any other representation and can be used to efficiently compute how the properties of the series change under mathematical operations such as multiplication or composition with a polynomial.

The **base** series are pure power series *i.e.* infinite series of the form

$$\sum_{i=0}^{\infty} a_i x^i.$$

The **target** series, defined implicitly via a recurrence structure and **candidate** series built from operations on **base** series, are generalised power series *i.e.* infinite series of the form

$$\sum_{j=0}^{\infty} b_j x^{j+s}$$

where s may be positive or negative, fractional or integral. Generalised power series subsume power series as a power series is just a generalised power series with $s=0$. Hence any representational scheme we devise for generalised power series can be used for power series also. Notice that the subscript on the coefficients of the **base** series is in step with the corresponding index but that of the **target** and **candidate** series is shifted by some offset value.

6.4.2 Representational Options

There are many ways we could represent an infinite series. Given the method by which we construct target series and our intended manipulations of base series some representations may prove easier to construct or easier to employ than others. Three contenders are: the *unmodified recurrence structure*, an *nth term representation* and a *case representation.*

6.4.3 Unmodified Recurrence Structure

The recurrence structure is an implicit definition of an infinite series. However, if we were to use this representation it would be very difficult to perform certain operations *e.g.* deriving the recurrence structure of the sum of two power series given recurrence structures of each one. So we reject using the raw recurrence structure.

6.4.4 nth Term Representation

A second possibility is to employ an *n*th term representation. Here we define the general form for the *n*th term of the series. In principle this may be derived by treating the recurrence structure as a difference equation and solving for the dependent variable. What we obtain, however, is not necessarily the term with coefficient a_n in the power series or generalised power series as the *n*th term representation specifies the *n*th *non-zero* term. Again this presents difficulties when attempting to merge two infinite series together. So we reject the *n*th term representation too.

6.4.5 Case Representations

The essence of the problem is to represent an infinite series in such a way that the description may easily be derived from the recurrence structure and changes to the infinite series wrought by mathematical operations are easily computed. The representation we advocate achieves these objectives not by representing the infinite series directly but rather by representing its *properties*. These include

- the set of indices
- the set of subscripts of all non-zero coefficients
- the set of subscripts of positive coefficients and negative coefficients
- the magnitudes of the first n non-zero coefficients

The key feature of the **case** representation is that we split the properties up into separate issues **each of which can be characterized by a set of numbers**. This allows us to treat each property independently and construct the set from the the subscript i of the coefficients a_i.

We cannot represent such sets of numbers literally as all but the last property would require infinite sets. Instead we use an expression in modular arithmetic as the generator of the set.

6.4.5.1 Qmodular Arithmetic

Qmodular arithmetic is similar to standard modular arithmetic: in both cases a proposition in the modular arithmetic implicitly defines an equivalence class of values. The difference is that qmodular arithmetic is defined over the rationals whereas modular arithmetic is over the integers. This means we can construct sets containing fractional values rather than just integers.

Definition 6-1. Qmod

$$\forall x\text{:rational}, \forall y\text{:integer } y > 0, \forall z\text{:rational } 0 \leq z < y$$

$$x \text{ Qmod } y = z \quad \leftrightarrow \quad \exists\, n\text{:integer } x = ny + z$$

A proposition in qmodular arithmetic therefore defines a whole family of values for x corresponding to all possible choices for n. Hence in qmodular arithmetic

$$\{i\text{: } i \text{ Qmod } 1 = {}^1/_2\}$$

is satisfiable and defines the set of $\{i\} = \{\ldots, -{}^7/_2, -{}^5/_2, -{}^3/_2, -{}^1/_2, {}^1/_2, {}^3/_2, {}^5/_2, {}^7/_2, \ldots\}$.

It proves useful to partition the equivalence class by augmenting the qmodular proposition with an inequality. This allows us to define an equivalence class of rationals greater than some desired value whilst excluding all smaller members. For example, the generator $\{i\colon 2i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$ implicitly defines the set $\{i\} = \{1/2, 3/2, 5/2, 7/2, \ldots\}$. In our application the set of indices present in the **target** series has just this form: an equivalence class cropped to the right of some minimum (and possibly negative) value.

If we want to define a set containing some negative values we can do so by shifting to the left *e.g.* $\{i\colon i + 5/2 \text{ Qmod } 2 = 1 \wedge i + 5/2 \geq 0\}$ defines the set $\{i\} = \{-3/2, 1/2, 5/2, 9/2, \ldots\}$. Similarly, if we want to define a set starting at a higher value we shift to the right *e.g.* $\{i\colon i - 5/2 \text{ Qmod } 2 = 1 \wedge i - 5/2 \geq 0\}$ defines the set $\{7/2, 11/2, 15/2, \ldots\}$.

Generators may be manipulated according to the following rules:

Match Rule

Given a pair of generators $\{i\colon i \text{ Qmod } p = q \wedge i \geq \alpha\}$ and $\{j\colon j \text{ Qmod } p = q \wedge j \geq \beta\}$, let the first values they induce be i_1 and j_1 respectively. Then,

$$\{i\colon i \text{ Qmod } p = q \wedge i \geq \alpha\} \equiv \{j\colon j \text{ Qmod } p = q \wedge j \geq \beta\} \text{ iff } i_1 = j_1$$

Shift Rule

$$\{i+\alpha\colon i \text{ Qmod } p = q \wedge i \geq \beta\} \equiv \{j\colon j \text{ Qmod } p = (\alpha+q) \text{ Qmod } p \wedge j \geq (\alpha+\beta)\}$$

Conjunction Rule

Two generators can be conjoined such that the set they determine is the intersection of those induced by each generator alone.

$$\{i: i \text{ Qmod } p = q \wedge i \geq \alpha\} \wedge \{i: i \text{ Qmod } p' = q' \wedge i \geq \beta\}$$

$$\equiv \{i: i \text{ Qmod } \text{lcm}(p, p') = \text{rmdr}(q, q', p, p') \wedge i \geq \max(\alpha, \beta)\}$$

where "lcm" returns the least common multiple of its arguments and "rmdr" is defined recursively by

$$\text{rmdr}(q, q', p, p') = \begin{cases} \text{undefined} & \text{iff } q*q' > \text{lcm}(p, p') \\ 0 & \text{iff } q = 0 \wedge q' = 0 \\ \text{rmdr}((q-1) \text{ Qmod } p, (q'-1) \text{ Qmod } p', p, p') + 1 & \text{otherwise} \end{cases}$$

Appendix II defines qmodular arithmetic and some associated rewrite rules for expressions in it.

6.4.5.2 Examples of Case Representation

An infinite series may now be represented as a tuple of **cases** one for each property.

$$\langle \textit{index-case}, \textit{coefficient-case}, \textit{sign-case}, \textit{magnitude-case} \rangle$$

The first three cases are a disjunction of generators in qmodular arithmetic:

$$\langle \{v_{i_1}:c_{i_1}\} \vee \{v_{i_1}:c_{i_1}\} \vee \ldots,$$

$$\{v_{c_1}:c_{c_1}\} \vee \{v_{c_1}:c_{c_1}\} \vee \ldots,$$

$$\{v_{s_1}:c_{s_1}\} \vee \{v_{s_1}:c_{s_1}\} \vee \ldots,$$

$$\textit{magnitude-case} \rangle$$

where the v_i are the sets of values induced by the qmodular constraints, c_i. Disjunctive possibilities are necessary because it may not be possible to characterise a set of numbers using only one qmodular proposition. Far from being a hindrance, this can actually be an asset when we come to decompose the **case** representation of the **target** into combinations of **base** series and polynomials.

The *magnitude case* is somewhat different: it is a finite set and is stored as a literal sequence of numbers. The magnitude case of the **target** is computed from the recurrence relation. Those of the **base** series are retrieved from the usual *n*th term representation in mathematical handbooks. The cardinality of the *magnitude case* is user definable but an acceptable figure would be five. This is somewhat *ad hoc* but not unreasonable. Our intended application is to find closed form approximations. It would require a very pathological case for two series to share the same first five terms and thereafter diverge.

Case representation allows us to flexibly state the contingent existence of properties of an infinite series without resorting to expensive symbolic manipulation.

Below we present four examples of **case** representations of infinite series.

series: $\cos(x) = 1 - x^2/2! + x^4/4! - x^6/6! + \ldots$

indices: $\{i: i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

coeffs: $\{a_i: i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

signs: $\{+1: i \text{ Qmod } 4 = 0 \wedge i \geq 0\} \vee \{-1: \text{i Qmod } 4 = 2 \wedge i \geq 0\}$

mgtde: $\{1, {}^1/_2, {}^1/_{24}, {}^1/_{720}, {}^1/_{40320}\}$

series: $e^{-x} * \sin(x) = x - x^2 + x^3/3 - x^5/30 + x^6/90 - x^7/630 + \ldots$

indices: $\{i: i \text{ Qmod } 4 = 2 \wedge i \geq 0\} \vee \{i: i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

coeffs: $\{a_i: i \text{ Qmod } 4 = 2 \wedge i \geq 0\} \vee \{a_i: i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

signs: $\{+1: i \text{ Qmod } 8 = 1 \wedge i \geq 0\} \vee \{+1: i \text{ Qmod } 8 = 3 \wedge i \geq 0\} \vee$

$\{+1: i \text{ Qmod } 8 = 6 \wedge i \geq 0\} \vee \{-1: i \text{ Qmod } 8 = 2 \wedge i \geq 0\} \vee$

$\{-1: i \text{ Qmod } 8 = 5 \wedge i \geq 0\} \vee \{-1: i \text{ Qmod } 8 = 7 \wedge i \geq 0\}$

mgtde: $\{1, 1, {}^1/_3, {}^1/_{30}, {}^1/_{90}\}$

series: $\sin(x^{1/2}) = x^{1/2} - x^{3/2}/3! + x^{5/2}/5! - x^{7/2}/7! + \ldots$

indices: $\{i: i \text{ Qmod } 1 = 1/2 \wedge i \geq 0\}$

coeffs: $\{a_i: i \text{ Qmod } 1 = 0 \wedge i \geq 0\}$

signs: $\{+1: i \text{ Qmod } 2 = 0 \wedge i \geq 0\} \vee \{-1: i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

mgtde: $\{1, 1/6, 1/120, 1/5040, 1/362880\}$

series: $\cos(x)/x^2 = x^{-2} - 1/2! + x^2/4! - x^4/6! + \ldots$

indices: $\{i: i \text{ Qmod } 2 = 0 \wedge i \geq -2\}$

coeffs: $\{a_i: i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

signs: $\{+1: i \text{ Qmod } 2 = 0 \wedge i \geq 0\} \vee \{-1: i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

mgtde: $\{1, 1/2, 1/24, 1/720, 1/40320\}$

6.5 Step (2): Mapping Recurrence Structure to Target Series

Once we have a *recurrence structure* we have, in principle, sufficient information to determine the **target** series *i.e.* the series solutions to the original differential equation. However, we need to be able to extract this information in the form of our infinite series representation. This is almost immediate from the recurrence structure. If the two indicial indices, s' and s'', do not differ by an integer, the recurrence structure will yield two solutions to the differential equation. For each value of $s \in S$ the *index*, *coefficient-existence*, *sign* and *magnitude* **cases** may be obtained by following Algorithm 6-2.

Algorithm 6-2. Map Recurrence Structure to Case Representation of **target**

Input

- recurrence structure

  $\langle$*recurrence relation, selectors, indicial indices*$\rangle$ =

  $$\langle b_i = \sigma * f(i;s) * b_{i-n},\ \{\{k^{s'}{}_1, k^{s'}{}_2, \ldots, k^{s'}{}_N\}, \{k^{s''}{}_1, k^{s''}{}_2, \ldots, k^{s''}{}_M\}\}, \{s', s''\}\rangle$$

Method

- instantiate **cases** according to the following definitions:

  indices: $\{i+s\colon i \text{ Qmod } n = k^s{}_1\} \vee \{i+s\colon i \text{ Qmod } n = k^s{}_2\} \vee \ldots \vee \{i+s\colon i \text{ Qmod } n = k^s{}_N\}$

  coeffs: $\{b_i\colon i \text{ Qmod } n = k^s{}_1\} \vee \{b_i\colon i \text{ Qmod } n = k^s{}_2\} \vee \ldots \vee \{b_i\colon i \text{ Qmod } n = k^s{}_N\}$

  signs: $\{\sigma_i = \sigma^{i-k^s{}_1 \text{ Qdiv } n}\colon i \text{ Qmod } n = k^s{}_1\} \vee \{\sigma_i = \sigma^{i-k^s{}_2 \text{ Qdiv } n}\colon i \text{ Qmod } n = k^s{}_2\} \vee$

  $$\ldots \vee \{\sigma_i = \sigma^{i-k^s{}_N \text{ Qdiv } n}\colon i \text{ Qmod } n = k^s{}_N\}$$

  mgtde: $\{|b_{j_1}|, |b_{j_2}|, |b_{j_3}|, \ldots, |b_{j_r}|\}$

The *sign case* is perhaps the least obvious. It arises as each $k^s{}_j$ in $\{k^s\}$ may "bottom out" a recurrence relation. Hence if some coefficient is an integral multiple of n from any $k^s{}_j$, it will be non-zero or indeterminate. Otherwise it will be zero. If there is more than one $k^s{}_j$ in $\{k^s\}$, the *index, coefficient* and *sign cases* will contain multiple disjunctive possibilities. In fact, the implied infinite series contains as many arbitrary constants as there are $k^s{}_j$ in k^s and each one can bottom out a recurrence relation.

As the sign of some term can only ever be +1 or –1 we can rewrite the condition on the signs explicitly (see Appendix III for proof), the exact form depending on whether or not the sign of the recurrence relation is positive or negative.

If $\sigma = +1$

signs: $\{\sigma_i = +1 : i \text{ Qmod } n = k^s{}_1\} \vee$

$\{\sigma_i = +1 : i \text{ Qmod } n = k^s{}_2\} \vee$

$\vdots$

$\{\sigma_i = +1 : i \text{ Qmod } n = k^s{}_N\}$

and if $\sigma = -1$

signs: $\{\sigma_i = +1 : i \text{ Qmod } 2n = k^s{}_1\} \vee \{\sigma_i = -1 : i \text{ Qmod } 2n = n + k^s{}_1\} \vee$

$\{\sigma_i = +1 : i \text{ Qmod } 2n = k^s{}_2\} \vee \{\sigma_i = -1 : i \text{ Qmod } 2n = n + k^s{}_2\} \vee$

$\vdots \qquad\qquad \vdots$

$\{\sigma_i = +1 : i \text{ Qmod } 2n = k^s{}_N\} \vee \{\sigma_i = -1 : i \text{ Qmod } 2n = n + k^s{}_N\}$

So from a recurrence structure we have constructed a representation of the salient features of the **target** series: the condition for the existence of coefficients, their associated indices and signs. This information is sufficient to allow us to recognise signature-index equivalent series.

6.6 Conclusions

In summary, normal equations with analytic coefficients can be solved in terms of power series expansions. Non-normal equations with pseudo-analytic coefficients can be solved in terms of generalised power series. In the latter case, the technique can always be used to find **at least one** solution of $y'' + (q(x)/x)y' + (r(x)/x^2)\, y = 0$. Abstract analysis of the method supplies general formulae for computing the coefficients of the series solution almost immediately from the equation. This circumvents most of the symbolic manipulation required to build the recurrence structure.

Chapter 7

Closed Form Approximation

7.1 Introduction

Infinite series solutions extend the range of equations that can be solved symbolically. However, it is difficult to comprehend the qualitative properties of such solutions. This chapter describes **closed form approximation**, a technique for approximating the exact infinite series solution of a differential equation as a combination of the elementary mathematical functions we all learn at school. By doing so we hope to retain the best features of both qualitative and mathematical descriptions: namely the comprehensibility of qualitative descriptions and conciseness and predictive utility of mathematical descriptions. Thus **closed form approximation** is another example of a technique for creating behavioural descriptions at the approximate functional level.

7.1.1 Relationship to Previous Techniques

Closed form approximation follows a different route to functional approximation than **analytic abduction** and **equation parsing**. These techniques worked by abstracting the

original differential equation, reasoning with the abstracted equation and then mapping the results back to the mathematical level. **Closed form approximation**, in contrast, builds its approximation directly from the exact, albeit infinite series, solution. Consequently, it has the pleasing property that, should a closed form solution exist and be constructible from the set of known elementary functions, **closed form approximation** will find it and prefer it over all other alternatives. Otherwise, if no closed form solution exists, the best approximation to it, given what functions are known, is made.

Within our framework the **closed form approximation** technique is necessary because **analytic abduction** cannot deal with singular equations owing to limitations of the **QSIM** algorithm. Likewise, **equation parsing** will fail if the solution cannot be written exactly in closed form. In this sense **closed form approximation** extends the range of problems that can be handled under the umbrella of approximate functional reasoning. Moreover, for those equations amenable to both **closed form approximation** and **equation parsing**, both techniques yield the same result †, modulo arbitrary constants. All three techniques are needed to cover a broad range of problems as no one technique subsumes any other entirely.

7.1.2 Criteria for Success

Our approach to finding a closed form approximation will be to find a closed form expression whose series expansion is "close" to that of the exact series solution. To do this we need to make this notion of closeness precise. We define two abstraction mappings of infinite series: *signature index* abstraction and *coefficient sequence* abstraction. Two series, in the independent variable x, are *signature index* equivalent if they share precisely the same set of powers of x and corresponding coefficients have the same signs. Alternatively, two series are *coefficient sequence* equivalent if the magnitudes of corresponding coefficients are equal. By relaxing *coefficient sequence* equivalence to something weaker, we can define a notion of proximity between two infinite series by

1) insisting that they be *signature index* equivalent and

2) defining a metric between coefficient sequences and requiring this to be less than some threshold value.

† assuming the requisite series expansions and function clichés are known.

These concepts are made precise below.

If two series are literally identical then clearly we want the metric between their coefficient sequences to be zero.

7.1.3 Terminology

Recall the terminology introduced in *Chapter 6*. The exact infinite series solution of the differential equation is called the **target** series. Any series which is close to the **target**, in the sense outlined above, is a **candidate** series. In general there will be many **candidates**, each a different distance from the **target**. The closest of these is termed the **twin**. The task for closed form approximation is to construct the twin by

1) constructing a set of **candidates** and

2) picking the **candidate** closest to the **target**.

7.1.4 Plan

The rest of this chapter is organised as follows. §7.2 develops the concepts of *signature index* and *coefficient sequence* equivalence between infinite series. §7.3 describes how infinite series, in the **case** representation introduced in the last chapter, transform under various mathematical operations. By using these rules backwards §7.4 explains how an algorithm can be conceived for explaining the target as the interaction of closed form functions and finding the **twin**. Following this, §7.5 provides examples of **closed form approximation** which shows that the method is faithful, in the sense that it finds the correct closed form solution if one exists and is expressible given the functions that are known. The penultimate section describes the limitations of closed form approximation and shows how the technique may sometimes overlap with equation parsing. Finally, §7.7 summarizes the research contributions and suggests future directions for research.

7.2 Equivalences Between Infinite Series

In order to tell whether one infinite series is a valid approximation of another it is necessary to define a notion of equivalence between them. This section describes two notions of equivalence based on different abstractions of the infinite series. **Signature index** abstraction can be used to test whether two series share the same sequence of indices and the same pattern of sign alternation. **Coefficient sequence** abstraction can be used to define a measure of how far they are apart. Together they determine whether two series can be considered to be "close" and if so "how close". The **twin** is the closed form approximation closest, in this sense, to the **target**.

If two functions to agree *via* their infinite series corresponding terms should have the same:-

- *exponent*
- *sign*
- *magnitude*

For the infinite series arising as solutions to differential equations, it is not possible to guarantee that a closed form can be found whose infinite series expansion is identical to the **target** (recall that the very reason solution in series techniques were developed was because it was known that not all functions could be represented in closed form). Since strict equality cannot be guaranteed this suggests, if an approximate solution is to be found, that the insistence on equality of some characteristic must be weakened. The most sensible one to relax is the equality of the magnitudes of corresponding coefficients. By abstracting only over the magnitudes of coefficients, two series can be ensured to match at least up to their sequences of powers of the independent variable and their signs for corresponding terms. However it would be foolish to throw away *all* the information retained in the magnitudes of coefficients because many infinite series will have the same *signature index* abstraction. Hence we define a metric between the coefficient sequences of the **target** and **candidate** that will enable the **candidate** which is "closest" to **target** to be determined.

7.2.1 Signature-Index Equivalence

Signature-index abstraction (λ) is a mapping between generalised power series (and hence power series) which reduces all coefficients to unity whilst retaining sign and index information.

$$\lambda: \sum_{i=0}^{\infty} a_i x^{i+s} \longmapsto \sum_{i=0}^{\infty} a_i/|a_i| \, x^{i+s}$$

This is the structure a mathematician would perceive at a glance.

Definition 7-1: *signature index* equivalence

Two series are *signature index* equivalent ($=_{SI}$) iff they have the same *signature index* abstraction

$$s_1 =_{SI} s_2 \leftrightarrow \lambda(s_1) = \lambda(s_2)$$

Signature-index equivalence provides the minimal notion of "closeness" between series, but on its own is insufficient as many series have the same *signature-index* abstraction. It is possible to do better by considering the nature of the sequence of coefficients in two *signature-index* equivalent series.

7.2.2 Coefficient Sequence Equivalence & Comparability

Coefficient-sequence$^{(r)}$ abstraction, $\xi^{(r)}$, is a mapping which takes a generalised power series to a sequence of real numbers whose members are the absolute values of the first r non-zero coefficients from the series. Letting i_ℓ be the subscript and a_{i_ℓ} the value of the ℓth non-zero coefficient in the **base** series, $\xi^{(r)}$ can be defined as the map

$$\xi^{(r)}: \sum_{i=0}^{\infty} a_i x^{i+m} \longmapsto \{|a_{i_1}|, |a_{i_2}|, |a_{i_3}|, ..., |a_{i_r}|\}.$$

Using this map it is possible to define an equivalence between coefficient sequences.

Definition 7-2: *coefficient-sequence*$^{(r)}$ equivalence

Two series are *coefficient-sequence*$^{(r)}$ equivalent ($=_{CS^{(r)}}$) iff they have the same *coefficient-sequence*$^{(r)}$ abstraction.

$$s_1 =_{CS^{(r)}} s_2 \leftrightarrow \xi^{(r)}(s_1) = \xi^{(r)}(s_2)$$

True equality between two generalised power series, s_1 and s_2, is now the conjunction

$$s_1 = s_2 \leftrightarrow s_1 =_{SI} s_2 \wedge s_1 =_{CS^{(\infty)}} s_2$$

i.e. a combination of *signature-index* equivalence and infinite *coefficient-sequence* equivalence. By replacing true equality with these new equivalences it is possible to define a natural notion of series approximation simply by relaxing the insistence on *coefficient sequence* equivalence whilst retaining *signature index* equivalence.

Instead of *coefficient sequence* equivalence we introduce a notion of *coefficient sequence* comparability by defining a metric between coefficient sequences. This allows a "distance" between coefficient sequences to be computed. Two series will then be *coefficient sequence* comparable if this distance is less than some threshold value.

A requirement for the metric is that it should embrace the intuition that because these sequences arise as coefficients of successively larger powers of x in an infinite series which is an approximation to a function in a neighbourhood around $x=0$, sequences which have early termwise agreement should be regarded as closer than those with later termwise agreement. This is because the lower order terms of the series expansions in our library of standard functions make a larger contribution to the function than the higher orders ones, at least when $|x|<1$. A suitable weighted metric between finite sequences is

$$d^{(r)}(\xi^{(r)}(s_1), \xi^{(r)}(s_2)) = \sum_{\ell=1}^{r} 1/\ell^2 \left| |a_{i\ell}| - |b_{i\ell}| \right| / (1 + \left| |a_{i\ell}| - |b_{i\ell}| \right|)$$

With this metric it is possible to define *coefficient-sequence* equivalence as

$$s_1 =_{CS^{(r)}} s_2 \leftrightarrow d^{(r)}(\xi^{(r)}(s_1), \xi^{(r)}(s_2)) = 0$$

It is, of course, possible to define alternative metrics; the only restriction is that they should be true metrics in the sense that they must satisfy three conditions which capture the basic properties of the concept of distance [Copson 79], namely:

$$d^{(r)}(\xi^{(r)}(s_1), \xi^{(r)}(s_2)) = 0 \leftrightarrow \xi^{(r)}(s_1) = \xi^{(r)}(s_2)$$

$$d^{(r)}(\xi^{(r)}(s_1), \xi^{(r)}(s_2)) = d^{(r)}(\xi^{(r)}(s_2), \xi^{(r)}(s_1))$$

$$d^{(r)}(\xi^{(r)}(s_1), \xi^{(r)}(s_2)) \leq d^{(r)}(\xi^{(r)}(s_1), \xi^{(r)}(s_3)) + d^{(r)}(\xi^{(r)}(s_3), \xi^{(r)}(s_2))$$

For a proof that the above distance measure is indeed a metric see [Copson 79 p28] for an analogous example.

7.2.3 Necessity of Metrics

Given two coefficient sequences there are non-metrical ways of comparing them. For example, two sequences could be regarded as equivalent if their elements *grow by the same index* or, alternatively, have the *same order of magnitude*. However, in order to deduce these relations a formula for the general term of the series would be needed. Building such a formula from a recurrence structure would involve expensive symbolic manipulation. Proving an equivalence relation between two such general formulae would involve even more effort. The **case** representation of **target** series avoids this problem by maintaining coefficient sequences rather than constructing the general form for a coefficient. However, this means that the necessary information to draw *grow by same index* or have the *same order of magnitude* inferences is not available. Hence we must resort to a metrical approach.

For the purpose of constructing a **closed form approximation** to a **target** a metrical approach is, in any case, superior. If more than one *signature index* equivalent series to the **target** is discovered a metrical approach allows the best one to be picked by choosing that **closed form approximation** which is closest to the **target**. Had only the relations *grow by same index* or have *same order of magnitude* been proved, it would be impossible to judge which of two candidates is better.

7.3 Manipulation of Infinite Series

To recapitulate, the previous sections have described a way of representing the properties of an infinite series, a notion of *signature index* equivalence and a way of assigning a measure of distance between *coefficient sequences*. In this section we will investigate how to transform the **case** representation of infinite series under certain mathematical operations. Having done this it will be possible to find a closed form approximation to a **target** by "explaining" the **target** as a **base** series modified by one of these mathematical operations.

In general, it will be possible to transform more than one **base** series into a **candidate**. In such cases, the metric introduced in the previous section can be used to determine which of these is the best approximation (*i.e.* the **twin**).

The following sections describe the two kinds of mathematical operation we have investigated: composition or multiplication of a **base** series with a simple polynomial. As the **base** series are always the Maclaurin expansions of closed form functions the approximation eventually found is guaranteed to have a closed form. Hence, the following mathematical operations underpin the whole **closed form approximation** technique.

7.3.1 Composition with a Polynomial

Given the general forms for a **base** and **target** series the first task is to determine how the **base** series is transformed under composition with a simple polynomial to a **candidate** series and then the conditions for this **candidate** to match the **target**.

As **base** series are always power series and **target** series generalised power series, the composition operation is:

$$\sum_{i=0}^{\infty} a_i x^i \circ cx^m = \sum_{i=0}^{\infty} a_i c^i x^{im} = \sum_{j=0}^{\infty} b_j x^{j+s}$$

To begin, consider the **base** series. The **case** description of the **base** series will always take the form

Base Series

indices:	$\{i: i \text{ Qmod } p = q \wedge i \geq 0\}$	
coeffs:	$\{a_i: i \text{ Qmod } p = q \wedge i \geq 0\}$	
signs:	$\{+1: i \text{ Qmod } p = q \wedge i \geq 0\}$	(if all positive)
	$\{-1: i \text{ Qmod } p = q \wedge i \geq 0\}$	(if all negative)
	$\{+1: i \text{ Qmod } 2p = q \wedge i \geq 0\} \vee \{-1: i \text{ Qmod } 2p = p+q \wedge i \geq 0\}$	(if alternating)
mgtde:	$\{\lvert a_{i_1}\rvert, \lvert a_{i_2}\rvert, \lvert a_{i_3}\rvert, \ldots, \lvert a_{i_r}\rvert\}$	

Only one of the *sign cases* is appropriate for a given series. The symmetric alternating case is omitted because there is no **base** series in the library of standard forms (see Appendix V) whose signs alternate, beginning with –1. As **base** series are power series all indices are integral and bounded below by zero. Moreover, as the subscript on the non-zero coefficients is always in step with the index, the condition for *coefficient existence* is identical to that determining *index sequence*. The sequence of signs of the coefficients in any **base** series either do not alternate or do so in the pattern {+,–,+,–,+,–,...} and never anything more complicated such as {+,+,–,+,+,–,+,+,–,...}. Finally **base** series requiring disjunctive possibilities in the *index* or *coefficient existence* **case** descriptions are excluded from the **base** series library. This merely simplifies the kinds of interactions which must be considered when combining **base** series with other functions.

Although these restrictions may sound severe, the representation is sufficiently flexible to describe all the Maclaurin expansions found in standard mathematical handbooks *e.g.* [Spiegel 68]. Moreover, it is also important to realize that series with complicated sign pattern alternation and disjunctive **case** possibilities can be derived from this kind of **base** series under the mathematical operations described below. So complicated series have not been excluded from consideration, merely from our repertoire of **"base"** series.

The next step is to determine how the **case** description of the **base** transforms upon composition with cx^m. We proceed in three stages, first by finding the conditions for *index equivalence*, then *signature equivalence* and finally *magnitude optimality*. This sequence in fact mirrors the steps in the algorithm we introduce in § 7.4 for constructing a closed form approximation.

7.3.1.1 Index Equivalence under Composition

From the functional definition of composition, the generators must take the form:-

Base

indices: $\{i: i \text{ Qmod } p = q \wedge i \geq 0\}$

coeffs: $\{a_i: i \text{ Qmod } p = q \wedge i \geq 0\}$

Candidate

indices: $\{im: i \text{ Qmod } p = q \wedge i \geq 0\}$

coeffs: $\{a_i c^i: i \text{ Qmod } p = q \wedge i \geq 0\}$

Target

indices: $\{j+s: j \text{ Qmod } n = k \wedge j \geq 0\}$

coeffs: $\{b_j: j \text{ Qmod } n = k \wedge j \geq 0\}$

Theorem 7.1: condition for **candidate-target** index equivalence under composition

The **candidate** and **target** will share the same indices iff

$$mp = n \ \wedge \ mq = s+k$$

Proof:

The proof is by induction on the sequence of indices spawned by **candidate** and **target** generators. If the **candidate** is to match the **target** the two sets of indices must be the same *i.e.*

$$\{im: i \text{ Qmod } p = q \wedge i \geq 0\} \equiv \{j+s: j \text{ Qmod } n = k \wedge j \geq 0\}$$

Let the sequences induced by these generators be

$$\{i_1 m, i_2 m, i_3 m, ..., i_\ell m, ...\} \text{ and } \{j_1+s, j_2+s, j_3+s, ..., j_\ell+s, ...\}$$

For the generators to be equivalent, corresponding members of these sets must be equal. To prove this consider the induction scheme

$$P(1) \wedge \forall \ell\, [P(\ell) \rightarrow P(\ell+1)] \rightarrow \forall x\, P(x)$$

Base Case

The lowest i is i_1. From the definition of Qmod, $i_1 = q$. Similarly, the lowest j is $j_1 = k$. $\therefore$ for base cases to match, $i_1 m = j_1 + s$, and so

$$mq = s + k$$

Step Case

Assume the induction hypothesis, $P(\ell)$, *i.e.* $i_\ell m = j_\ell + s$. It must now be shown that if this is true, $i_{\ell+1} * m = j_{\ell+1} + s$.

From the definition of Qmod $i_{\ell+1} - i_\ell = p$ and $j_{\ell+1} - j_\ell = n$.

$$\therefore\ i_{\ell+1} * m = (p + i_\ell) * m$$

and

$$j_{\ell+1} + s = n + j_\ell + s.$$

But the induction hypothesis states that $i_\ell m = j_\ell + s$. $\therefore\ i_{\ell+1} * m = j_{\ell+1} + s$ iff

$$mp = n$$

□

This guarantees *index* equivalence but not *signature* equivalence. To achieve the latter it is necessary to examine how the signs of the coefficients transform under composition with cx^m.

7.3.1.2 Signature Equivalence under Composition

The precise form of the transformation will depend on whether all the signs of the coefficients of the series are the same or alternate in some way, whether the indices are odd or even and whether c in the polynomial cx^m is positive or negative. For each possibility, the *sign case* of the **base** is first mapped to that of the **candidate**. Then, using the relationship between **base** and **target** parameters given by Theorem 7.1 and the rewrite rules for generators derived in Appendix IV, the *sign case* of the **candidate** is written in terms of the **target**'s parameters. Finally, this is checked, by syntactic equivalence, to determine whether or not the resulting generator matches that of the **target**.

The table below summarizes the findings of Appendix IV which show the mappings between the *sign cases* of **base** and **candidate** written in terms of **target** parameters. If m is not an integer multiple of $\frac{1}{2}$ it will be necessary to scale the congruence so that $2m$ is an integer.

Rule ◦1: coefficients all positive, $c > 0$

signs: $\{+1: i \text{ Qmod } p = q\} \longmapsto \{+1: j \text{ Qmod } n = k\}$

Rule ◦2: coefficients all positive, $c < 0$

signs: $\{+1: i \text{ Qmod } p = q\} \longmapsto$

$$\{+1: j \text{ Qmod } \text{lcm}(n,2m) = \text{rmdr}(k, (-s) \text{ Qmod } 2m, n, 2m)\} \vee$$
$$\{-1: j \text{ Qmod } \text{lcm}(n,2m) = \text{rmdr}(k, (m-s) \text{ Qmod } 2m, n, 2m)\}$$

Rule ◦3: coefficients alternate, $c > 0$

signs: $\{+1: i \text{ Qmod } 2p = q\} \quad \{+1: j \text{ Qmod } 2n = k\}$
$\longmapsto$
$\{-1: i \text{ Qmod } 2p = p+q\} \quad \{-1: j \text{ Qmod } 2n = n+k\}$

Rule ∘4a: coefficients alternate, $c < 0$, even(p), even(q)

signs: $\{+1: i \text{ Qmod } 2p = q\}$

$\{-1: i \text{ Qmod } 2p = p+q\}$

$\longmapsto$

$\{+1: j \text{ Qmod } \text{lcm}(2n,2m) = \text{rmdr}(k, (-s) \text{ Qmod } 2m, 2n, 2m)\}$

$\{-1: j \text{ Qmod } \text{lcm}(2n,2m) = \text{rmdr}(n+k, (-s) \text{ Qmod } 2m, 2n, 2m)\}$

Rule ∘4b: coefficients alternate, $c < 0$, even(p), odd(q)

signs: $\{+1: i \text{ Qmod } 2p = q\}$

$\{-1: i \text{ Qmod } 2p=p+q\}$

$\longmapsto$

$\{+1: j \text{ Qmod } \text{lcm}(2n,2m) = \text{rmdr}(n+k, (m-s) \text{ Qmod } 2m, 2n, 2m)\}$

$\{-1: j \text{ Qmod } \text{lcm}(2n,2m) = \text{rmdr}(k, (m-s) \text{ Qmod } 2m, 2n, 2m)\}$

Rule ∘4c: coefficients alternate, $c < 0$, odd(p), even(q)

signs: $\{+1: i \text{ Qmod } 2p = q\}$

$\{-1: i \text{ Qmod } 2p=p+q\}$

$\longmapsto$

$\{+1: j \text{ Qmod } \text{lcm}(2n,2m) = \text{rmdr}(k, (-s) \text{ Qmod } 2m, 2n, 2m)\}$

$\{+1: j \text{ Qmod } \text{lcm}(2n,2m) = \text{rmdr}(n+k, (m-s) \text{ Qmod } 2m, 2n, 2m)\}$

Rule ∘4d: coefficients alternate, $c < 0$, odd(p), odd(q)

signs: $\{+1: i \text{ Qmod } 2p = q\}$

$\{-1: i \text{ Qmod } 2p=p+q\}$

$\longmapsto$

$\{-1: j \text{ Qmod } \text{lcm}(2n,2m) = \text{rmdr}(k, (m-s) \text{ Qmod } 2m, 2n, 2m)\}$

$\{-1: j \text{ Qmod } \text{lcm}(2n,2m) = \text{rmdr}(n+k, (-s) \text{ Qmod } 2m, 2n, 2m)\}$

If the conditions for *index* equivalence are met and the *sign cases* of the **target** series match those of the **candidate** series, *i.e.* the right hand side of the above rules, an index m and the sign of the coefficient c in cx^m will have been determined such that the **target** and **candidate** are *signature index* equivalent. All that remains to be done now is to determine a magnitude for c such that the first terms of the series agree. Once this is done the distance between the **candidate** and the **target** can be computed. If the two series are truly equal this distance will be zero.

7.3.1.3 Magnitude Optimality under Composition

	Base	**Candidate**	**Target**
coeffs:	$\{a_i: i \text{ Qmod } p = q\}$	$\{a_i c^i: im \text{ Qmod } mp = mq\}$	$\{b_j: j \text{ Qmod } n = k\}$
mgtde:	$\{\|a_{i_1}\|, \|a_{i_2}\|, \|a_{i_3}\|, ..., \|a_{i_r}\|\}$	$\alpha*\{\|a_{i_1}c^{i_1}\|, \|a_{i_2}c^{i_2}\|, \|a_{i_3}c^{i_3}\|, ..., \|a_{i_r}c^{i_r}\|\}$	$\beta*\{\|b_{j_1}\|, \|b_{j_2}\|, \|b_{j_3}\|, ..., \|b_{j_r}\|\}$

To fix the scaling of the series and the magnitude of c it is necessary to equate leading terms, $\alpha*|a_{i_1}c^{i_1}| = \beta*|b_{j_1}|$ and $\alpha*|a_{i_2}c^{i_2}| = \beta*|b_{j_2}|$. The remaining elements of the *coefficient sequence*$^{(r)}$ abstraction of the **candidate** are then computed using this value for c. Although a value for c could, in principle, be derived by equating any two a_ic^i and b_i $(i>0)$, it is best to use $a_{i_1}c^{i_1}$ and b_{j_1} as we want to ensure that the leading powers, which make a proportionately larger contribution to the function, agree. This is explained more fully in §7.4.

7.3.2 Multiplication with a Polynomial

The next operation to be investigated is how the **cases** of the **base** series transform when it is multiplied with the polynomial cx^m. Mathematically, the operation is as follows:

$$\sum_{i=0}^{\infty} a_i x^i * cx^m = \sum_{i=0}^{\infty} a_i c x^{i+m} = \sum_{j=0}^{\infty} b_j x^{j+s}$$

As above we will compute the *index, coefficient-existence, sign* and *magnitude* cases of the **candidate** series in the middle given those of the **base** series on the left. Again we will first achieve *index* equivalence, then *signature* equivalence and finally *magnitude* optimality, reflecting the stages in the **closed form approximation** algorithm.

7.3.2.1 Index Equivalence under Multiplication

Using the mathematical definition of multiplication, the index and coefficient-existence generators must take the form:-

Base

indices: $\{i: i \text{ Qmod } p = q \wedge i \geq 0\}$

coeffs: $\{a_i: i \text{ Qmod } p = q \wedge i \geq 0\}$

Candidate

indices: $\{i+m: i \text{ Qmod } p = q \wedge i \geq 0\}$

coeffs: $\{a_i c: i \text{ Qmod } p = q \wedge i \geq 0\}$

Target

indices: $\{j+s: j \text{ Qmod } n = k \wedge j \geq 0\}$

coeffs: $\{b_j: j \text{ Qmod } n = k \wedge j \geq 0\}$

Theorem 7.2: condition for **candidate-target** index equivalence under multiplication

The sequence of indices of **candidate** and **target** will be the same iff

$$p = n \ \wedge \ m = s+k-q$$

Proof:

The proof is by induction on the sequence of indices spawned by **candidate** and **target** generators. If the **candidate** is to match the **target** the two sets of indices must be the same *i.e.*

$$\{i+m: i \text{ Qmod } p = q \wedge i \geq 0\} \equiv \{j+s: j \text{ Qmod } n = k \wedge j \geq 0\}$$

Let the sequences induced by these generators be

$$\{i_1+m, i_2+m, i_3+m, ..., i_\ell+m, ...\} \text{ and } \{j_1+s, j_2+s, j_3+s, ..., j_\ell+s, ...\}$$

For the generators to be equivalent, corresponding members of these sets must be equal. To prove this consider the induction scheme

$$P(1) \wedge \forall \ell\, [P(\ell) \rightarrow P(\ell+1)] \rightarrow \forall x\, P(x)$$

Base Case

The lowest i is i_1. From the definition of Qmod, $i_1 = q$. Similarly, the lowest j is $j_1 = k$. $\therefore$ for base cases to match, $i_1 + m = j_1 + s$, and so

$$m = s + k - q$$

Step Case

Assume the induction hypothesis, $P(\ell)$, *i.e.* $i_\ell + m = j_\ell + s$. It must now be shown that, if this is true, $i_{\ell+1} + m = j_{\ell+1} + s$. From the definition of Qmod $i_{\ell+1} - i_\ell = p$ and $j_{\ell+1} - j_\ell = n$.

$$\therefore\ i_{\ell+1} + m = p + i_\ell + m$$

and

$$j_{\ell+1} + s = n + j_\ell + s.$$

But the induction hypothesis states that $i_\ell + m = j_\ell + s$. $\therefore\ i_{\ell+1} + m = j_{\ell+1} + s$ iff

$$p = n$$

□

This only guarantees *index* equivalence. To achieve *signature* equivalence it is necessary to examine how the signs of the coefficients transform under multiplication with cx^m.

7.3.2.2 Signature Equivalence under Multiplication

The precise form of the transformation will depend on whether the signs of the coefficients of the series are the same or alternate in some way and what the sign of c in the polynomial cx^m is.

Rule *1: coefficients all positive, $c > 0$

signs: $\{+1: i \text{ Qmod } p = q\} \longmapsto \{+1: j \text{ Qmod } n = k\}$

Rule *2: coefficients all positive, $c < 0$

signs: $\{+1: i \text{ Qmod } p = q\} \longmapsto \{-1: j \text{ Qmod } n = k\}$

Rule *3: coefficients alternate, $c > 0$

signs:
$$\begin{array}{ll} \{+1: i \text{ Qmod } 2p = q\} & \{+1: j \text{ Qmod } 2n = k\} \\ \{-1: i \text{ Qmod } 2p = p+q\} & \{-1: j \text{ Qmod } 2n = n+k\} \end{array} \longmapsto$$

Rule *4: coefficients alternate, $c < 0$

signs:
$$\begin{array}{ll} \{+1: i \text{ Qmod } 2p = q\} & \{-1: j \text{ Qmod } 2n = k\} \\ \{-1: i \text{ Qmod } 2p = p+q\} & \{+1: j \text{ Qmod } 2n = n+k\} \end{array} \longmapsto$$

Again, if the conditions for *index* equivalence are met and the **target** series matches one of the right hand side forms for the **candidate**, then an index m and the sign of c in cx^m will have been discovered such that the **candidate** is *signature index* equivalent to the **target**.

7.3.2.3 Magnitude Optimality under Multiplication

	Base	**Candidate**	**Target**
coeffs:	$\{a_i: i \, Q mod \, p = q\}$	$\{a_i c: im \, Q mod \, mp = mq\}$	$\{b_j: j \, Q mod \, n = k\}$
mgtde:	$\{\|a_{i_1}\|, \|a_{i_2}\|, \|a_{i_3}\|, ..., \|a_{i_r}\|\}$	$\alpha^*\{\|a_{i_1}c\|, \|a_{i_2}c\|, \|a_{i_3}c\|, ..., \|a_{i_r}c\|\}$	$\beta^*\{\|b_{j_1}\|, \|b_{j_2}\|, \|b_{j_3}\|, ..., \|b_{j_r}\|\}$

To fix the scaling of the series a is equated with β. To fix the magnitude of c, $|a_{i_1}c|$ is equated with $|b_{j_1}|$. The remaining elements of the *coefficient sequence*$^{(r)}$ abstraction of the **candidate** are then computed using this value for c. We elaborate on this in the following section.

7.4 Closed Form Approximation Algorithm

Chapter 6 and §7.2 established the theoretical machinery for representing, proving equivalences between and manipulating infinite series. In this section these results are used to construct an algorithm for automatically generating closed form approximations. Four distinct stages are involved:

1) given an input differential equation, determine the *recurrence structure* of its solutions

2) given a *recurrence structure*, transform this to a **case** representation of an infinite series (the **target** series)

3) given a **target** series, find a set of closed form approximations (the **candidates**)

4) determine which of the **candidates** is closest to the **target** (*i.e.* determine the **twin**)

7.4.1 Step (1) Solution in Series

The first step was explained in the last Chapter. The essential point to note is that it is possible to use the general form for a recurrence relation to circumvent most of the expensive symbolic manipulation seen in textbook examples of the solution in series method.

7.4.2 Step (2) Mapping Recurrence Structure to Target Series

Again, this was explained fully in the last Chapter. The key innovation was the use of a **case** representation for infinite series. This partitions the properties of the series into the set of indices, the set of subscripts of non-zero coefficients, the set of signs and the set of magnitudes of the first five terms. The first three sets, which are infinite, are determined by generators in qmodular arithmetic. The last one is a literal finite set. The advantage of a **case** representation is that it is easily derived from the recurrence structure and facilitates the computation of the effects of various mathematical operations on the series.

7.4.3 Step (3) Finding a Set of Closed Form Approximations to Target

The standard solution in series method stops upon arrival at an infinite series solution. The **closed form approximation** technique goes further. Its purpose is to find a closed form approximation to the (exact) solution so that its qualitative properties are more readily apparent.

To do this it is necessary to find a decomposition of the **target** series into a compound expression involving **base** series and polynomials such that the composite expression is both *signature index* equivalent and *coefficient sequence* comparable to the **target**. As all the **base** series in the standard library (see Appendix V) have a known closed form this guarantees that the composite expression also has a closed form. Moreover, if multiple **candidates** are found it is easy to identify which is the best approximation by finding the

one which is "closest", in a metrical sense, to the **target**. If it so happened that the **target** series could be expressed exactly in a closed form constructible from the set of known **base** series, then the **closed form approximation** technique is guaranteed to find it and the distance between **candidate** and **target** will then be zero.

Recall the syntax for a recurrence structure defined in *Chapter 6*. $\{k^s\}$ is the set of *selectors* for the indicial index s. These are the subscripts of the non-zero and indeterminate coefficients of the **target** series which are less than n, the spacing between indices. The algorithm for constructing the set of **candidates** is as follows:

Algorithm 7-1. Explain **target** series

Input

- the case representation of a **target** series
- the cardinality of $\{k^s\}$

Method

3i) split **target** into as many series as there are k^s_j in $\{k^s\}$

for each such series,

for each **base** series in library do

3ii) test whether **base** $\stackrel{?}{=}$ **target**

3iii) and test whether $\exists\ c, m$: **base** $\circ\ cx^m \stackrel{?}{=}$ **target**

3iv) and test whether $\exists\ c, m$: **base** $*\ cx^m \stackrel{?}{=}$ **target**

The rationale behind the first step is that each distinct k^s_j introduces a new arbitrary constant into the series. Each **base** series can be uniformly scaled by multiplying it by an arbitrary constant. Thus in order to recognise a scaled **base** series we only want one arbitrary constant. Hence any **target** series containing multiple arbitrary constants needs to be partitioned into a sum of **targets** each containing just one arbitrary constant. This is very simple as the disjunctions are already explicit and we break the series at these points. The only complication is that extra coefficient magnitudes must be evaluated to ensure that

the *mgtde case* of each new series has a full complement of entries (*i.e.* the first r non-zero coefficient magnitudes for each new series). These may easily be calculated using the recurrence relation.

The next stage is to test whether the **base** series matches the **target** directly or whether a c and m can be found such that the **base** composed or multiplied with cx^m can match the **target**. Each test terminates in a call to the matcher which simply tests for syntactic equivalence (if two sequences are the same their generators can always be rewritten to the same form).

In either case the processes involved are similar. First an m is sought such that Theorem 7.1 or Theorem 7.2 is satisfied. If a suitable value is found, this guarantees that the **candidate** can be made *index* equivalent to the **target**. Next a sign for c is sought which renders the **candidate** and **target** *signature* equivalent too. This procedures constructs a set of **candidates** which are *signature-index* equivalent to **target**. Finally the magnitude of c is optimised to ensure maximum agreement between coefficient sequences by enforcing equality with the leading terms of the series.

7.4.4 Step (4) Finding the Best Closed Form Approximation

If there are multiple **candidates** which are *signature index* equivalent to **target** it is necessary to determine which of them is the best approximation *i.e.* which one is the **twin**. This can be done *via* Algorithm 7-2 as follows:

Algorithm 7-2. Determine the **twin**

Input

- the case descriptions of a set of **candidates**
- the case description of the **target**

Method

for each **candidate-target** pair **do**

- evaluate $d^{(5)}(\xi^{(5)}(\textbf{candidate}), \xi^{(5)}(\textbf{target}))$
- return the candidate for which this metric is a minimum

The *mgtde case* of an infinite series is the sequence of absolute values of the first r non-zero coefficients in the series. At this stage, each **target** series contains at most one arbitrary constant (remember the first step was to partition a target containing multiple arbitrary constants into a set of series each containing just one). So each term in the *mgtde case* is in fact multiplied by this unique symbolic constant (β say).

So far, the properties of the **base** and **target** have been dealt with independent of actual coefficient magnitude information. However, even if a **base** were selected which was of an identical functional form to the real **target** series it would still be necessary to rescale it to agree with this constant β. The final stage of the matching process is therefore to rescale the **candidate** series (by fixing α) and identify a value for the constant c. Let the candidate be uniformly scaled by the constant α and the **target** by the constant β. So, for example, for composition, the **mgtde** cases are as follows:

	Base	**Candidate**	**Target**
mgtde:	$\{\|a_{i_1}\|, \|a_{i_2}\|, \|a_{i_3}\|, ..., \|a_{i_r}\|\}$	$\alpha*\{\|a_{i_1}c^{i_1}\|, \|a_{i_2}c^{i_2}\|, \|a_{i_3}c^{i_3}\|, ..., \|a_{i_r}c^{i_r}\|\}$	$\beta*\{\|b_{j_1}\|, \|b_{j_2}\|, \|b_{j_3}\|, ..., \|b_{j_r}\|\}$

For the *mgtde case* of the **candidate** to match that of the **target**, corresponding elements of the *mgtde case* will be equal. In principle, a value for c can therefore be computed by equating corresponding elements of each *mgtde case* and solving the simultaneous set of equations for c:

$$\alpha*\{|a_{i_1}c^{i_1}|, |a_{i_2}c^{i_2}|, |a_{i_3}c^{i_3}|, ..., |a_{i_r}c^{i_r}|\} = \beta*\{|b_{j_1}|, |b_{j_2}|, |b_{j_3}|, ..., |b_{j_r}|\}$$

In the ideal case when the **candidate** and **target** are truly equal a unique value for $|c|$ is obtained by solving any one of the equations. However, in general, the set of simultaneous equations will be ill-posed *i.e.* it will not be possible to find a unique value for $|c|$ such that all the equations simultaneously hold. In this case it will only be possible to find an approximate solution. A value for $|c|$ could be derived by equating any pair of $\alpha*|a_{i_\ell}c^{i_\ell}|$ and $\beta*|b_{j_\ell}|$ ($i>0$). However, it is best to use $a_{i_1}c^{i_1}$ and b_{j_1} to ensure that the leading powers, which make a proportionately larger contribution to the function, agree.

By equating leading terms:

$$a^*|a_{i_1}c^{i_1}| = \beta^*|b_{j_1}| \quad \wedge \quad a^*|a_{i_2}c^{i_2}| = \beta^*|b_{j_2}|$$

rough values for a and c may be determined.

$$|c| = [(|a_{i_1}| * |b_{j_2}|)/(|a_{i_2}| * |b_{j_1}|)]^{1/(i_2 - i_1)} \quad \wedge \quad a = \beta^*(|b_{j_1}| + |b_{j_2}|)/(|a_{i_1}c^{i_1}| + |a_{i_2}c^{i_2}|)$$

The remaining elements of the *mgtde case* of the **candidate** are then computed using this value for c: the ℓth element being $|a_{i_\ell}c^{i_\ell}|$ where a_ℓ is the ℓth element of the *mgtde case* of the **base**. These values are then used to determine the distance between the **candidate** and the **target**.

In the case of a **base** series multiplied with a polynomial the rescaling parameter can be eliminated by absorbing its effects into c.

	Base	**Candidate**	**Target**
coeffs:	$\{a_i: i\ Q mod\ p = q\}$	$\{a_i c: im\ Q mod\ mp = mq\}$	$\{b_j: j\ Q mod\ n = k\}$
mgtde:	$\{\|a_{i_1}\|, \|a_{i_2}\|, \|a_{i_3}\|, ..., \|a_{i_r}\|\}$	$a^*\{\|a_{i_1}c\|, \|a_{i_2}c\|, \|a_{i_3}c\|, ..., \|a_{i_r}c\|\}$	$\beta^*\{\|b_{j_1}\|, \|b_{j_2}\|, \|b_{j_3}\|, ..., \|b_{j_r}\|\}$

In other words we can freely set $a = \beta$ and equate leading terms, $|a_{i_1}c| = |b_{j_1}|$, to determine $|c|$.

$$|c| = |b_{j_1}|/|a_{i_1}| \quad \wedge \quad a = \beta$$

7.5 Examples

In this section we present three examples of **closed form approximation**. The first demonstrates the behaviour of the **closed form approximation** algorithm when no exact closed form solution exists. Its interesting feature is the way it selects the best closed form approximation from competing candidates. The second and third examples testify to the faithfulness of the algorithm. These are both cases where an exact closed form solution does exist and show that the **closed form approximation** algorithm both finds and prefers the "correct" solution over all other candidates provided the requisite functions are known to the library. Moreover, these examples are deliberately chosen from standard textbooks to facilitate comparison between their laborious method of solution and our algorithmic method.

7.5.1 Approximation when Closed Form Solution Impossible

Step (1) Solution in Series

Consider the equation

$$2x^2\, d^2y/dx^2 - x\, dy/dx + (1-x^2)y = 0$$

1i) put equation in normal form

$$d^2y/dx^2 + (1/x)(-1/2)\, dy/dx + (1/x^2)(1/2 + (-1/2)x^2)y = 0$$

1ii) verify conditions for solution in series satisfied

$q(x) = -1/2$ and $r(x) = 1/2 + (-1/2)x^2$ are expandable as convergent power series at $x=0$ (trivial because both are finite polynomials)

1iii) compute expansions for $q(x)$ and $r(x)$

$$q(x) = -1/2\, x^0 + 0x + 0x^2 + \ldots \equiv q_0 = -1/2, \quad q_i = 0\ \forall\, i>0$$

$$r(x) = 1/2\, x^0 + 0x + (-1/2)x^2 + 0x^3 \ldots \equiv r_0 = 1/2, r_1 = 0, r_2 = -1/2, r_i = 0\ \forall\, i>2$$

1iv) evaluate $q(0)$ and $r(0)$, set up and solve indicial equation

$$q(0) = q_0 = -1/2$$

$$r(0) = r_0 = 1/2$$

indicial equation: $s(s-1) + q(0)s + r(0) = 0$ becomes

$$s^2 - {}^3/_2\, s + {}^1/_2 = 0 \Rightarrow s = {}^1/_2 \vee s = 1$$

1v) compute general form for recurrence relation

$$b_i = -(1/I(i+s)) \sum_{j=0}^{i-1} [(j+s)\, q_{i-j} + r_{i-j}]\, b_j, \quad i \geq 1$$

where $I(x) = x(x-1) - {}^1/_2\, x + {}^1/_2 = (x-{}^1/_2)(x-1)$

$$b_i = -(1/(i+s-{}^1/_2)(i+s-1)) \sum_{j=0}^{i-1} [(j+s)\, q_{i-j} + r_{i-j}]\, b_j, \quad i \geq 1$$

1vi) simplify recurrence relation

Consider the implications of the summation $\sum_{j=0}^{i-1}$. The smallest q_{i-j} in the sum is q_1. But $q_i = 0\ \forall\ i>0 \therefore$ the term in q_{i-j} makes no contribution. Conversely, the term in r_{i-j} does make a contribution as the smallest r_{i-j} in the sum is r_1 which precedes a non-zero term, r_2. However, $r_i = 0\ \forall\ i>2$ so r_2 is the *only* non-zero term in the summation. $r_{i-j} = r_2$ iff $i-j=2$. Therefore, the recurrence relation simplifies to

$$b_i = -(1/(i+s-{}^1/_2)(i+s-1))\, r_2\, b_{i-2}, \quad i \geq 2$$

which, as $r_2 = {}^{-1}/_2$ becomes

$$b_i = +1 * {}^1/_{2(i+s-{}^1/_2)(i+s-1)} * b_{i-2}, \quad i \geq 2$$

1vii) for each indicial index, identify corresponding selectors

Use the original unsimplified recurrence relation to evaluate for each *indicial index* s each $b_i\ (0 < i < 2)$. For this problem b_1 is the only coefficient to evaluate.

For $s={}^1/_2, i=1$

$$b_1 = -({}^1/_{({}^1/_2)})\, [(0+{}^1/_2)\, q_1 + r_1]\, b_0 = 0$$

For $s=1, i=1$

$$b_1 = -({}^1/_{({}^3/_2)})\, [(0+1)\, q_1 + r_1]\, b_0 = 0$$

1viii) create recurrence structure

Collecting the above results together, the recurrence structure is

⟨*recurrence relation, selectors, indicial indices*⟩ =

$$\langle b_i = +1 * {}^{1}/_{2(i+s-1/2)(i+s-1)} * b_{i-2}, \{k^{(1/2)}=\{0\}, k^{(1)}=\{0\}\}, \{s={}^{1}/_{2}, s=1\}\rangle$$

Step (2) Mapping Recurrence Structure to Target Series

The *recurrence relation* implicitly defines two target series, one for $s={}^{1}/_{2}$ and the other for $s=1$. As the sign of the recurrence relation is positive this means that all coefficients will have the same sign. Taking the $s={}^{1}/_{2}$ case first, $s={}^{1}/_{2}$, $n=2$, $k=0$ and applying the mapping between recurrence structure and **target** series (Appendix III) $\mathbf{target_1}$ is obtained.

$\mathbf{target_1}$

indices: $\{j+{}^{1}/_{2}: j \text{ Qmod } 2 = 0 \wedge j \geq 0\}$

coeffs: $\{j: j \text{ Qmod } 2 = 0 \wedge j \geq 0\}$

signs: $\{+1: j \text{ Qmod } 2 = 0 \wedge j \geq 0\}$

mgtde: $b_0 * \{1, {}^{1}/_{6}, {}^{1}/_{168}, {}^{1}/_{11088}, {}^{1}/_{1330560}\}$

Similarly, for the other case: $s=1$, $n=2$, $k=0$, $\mathbf{target_2}$ is

---

$\mathbf{target_2}$

indices: $\{j+1: j \text{ Qmod } 2 = 0 \wedge j \geq 0\}$

coeffs: $\{j: j \text{ Qmod } 2 = 0 \wedge j \geq 0\}$

signs: $\{+1: j \text{ Qmod } 2 = 0 \wedge j \geq 0\}$

mgtde: $b_0 * \{1, {}^{1}/_{10}, {}^{1}/_{360}, {}^{1}/_{28080}, {}^{1}/_{3818880}\}$

We will ignore $\mathbf{target_2}$ from this point onwards as the procedure is entirely analogous.

Step (3) Finding a Set of Closed Form Approximations to Target

3i) split target into as many series as there are k^s_j in $\{k^s\}$

in this case trivial as there is only one selector for each index

for each such series,

for each base series in library do

3ii) test whether base $\stackrel{?}{=}$ target

none of the bases in the library directly match **target$_1$**

3iii) and test whether $\exists\, c, m$: base $\circ\, cx^m \stackrel{?}{=}$ target

Condition is: $mp=n \wedge mq=s+k$
Instantiates to: $mp=2 \wedge mq=1/2$

fails for **target$_1$**

3iv) and test whether $\exists\, c, m$: base $*\, cx^m \stackrel{?}{=}$ target

Condition is: $p=n \wedge m=s+k-q$ (theorem 7.2)
Instantiates to: $p=2 \wedge m=1/2-q$

succeeds for **target$_1$** with $n=2$, $k=0$, $s=1/2$

Fix m

Theorem 7.2 is satisfied by the following **base** series with suitable values for p, q and m:

p	q	m	**functions**
2	0	$1/2$	$\cos x \quad \sec x \quad \cosh x \quad \operatorname{sech} x \quad \exp x^2$
2	1	$-1/2$	$\sin x \quad \tan x \quad \arcsin x \quad \arctan x \quad \sinh x \quad \tanh x \quad \operatorname{arcsinh} x \quad \operatorname{arctanh} x$

Fix sign c

The *sign case* of **target$_1$** is

signs: $\{+1 : j \text{ Qmod } 2 = 0 \wedge j \geq 0\}$.

We now filter the **base** series retaining only those which, for some sign of c and under multiplication with cx^m, can be made signature equivalent to **target**$_1$. These are shown below together with the corresponding sign for c and the rule under which the mapping is carried out.

p	q	m	**functions**	**sign**(c)	**rule**
2	0	$^1/_2$	$\sec x \quad \cosh x \quad \exp x^2$	$c>0$	*1
2	1	$-^1/_2$	$\tan x \quad \arcsin x \quad \sinh x \quad \operatorname{arctanh} x$	$c>0$	*1

Each of these **base** series can therefore be made signature-index equivalent to **target**$_1$. However, in order to determine which is the best approximation (*i.e.* the **twin**) it is necessary to examine the relative magnitudes of the coefficients in the **candidate** and **target** series.

Fix magnitude of c

The magnitude case of **target**$_1$ is:

mgtde: $b_0*\{1, {}^1/_6, {}^1/_{168}, {}^1/_{11088}, {}^1/_{1330560}\}$

Equate $a*\{|a_{i_1}c|, |a_{i_2}c|, \ldots, |a_{i_5}c|\}$ with $b_0*\{1, {}^1/_6, {}^1/_{168}, {}^1/_{11088}, {}^1/_{1330560}\}$ and compute approximate values for $|c|$ and a from

$$|c| = |b_{j_1}|/|a_{i_1}| \quad \wedge \quad a = b_0$$

Recall the notation: i_ℓ is the subscript of the ℓth non-zero coefficient in the **base** series and it's value is a_{i_ℓ}

p	q	m	**functions** $f(x)$	$\lvert c \rvert$	a	**CFA**
2	0	$^1/_2$	$\sec x \quad \cosh x \quad \exp x^2$	1	b_0	$b_0 x^{1/2} f(x)$
2	1	$-^1/_2$	$\tan x \quad \arcsin x \quad \sinh x \quad \operatorname{arctanh} x$	1	b_0	$b_0 x^{-1/2} f(x)$

Hence it is possible to evaluate the *magnitude cases* of the candidates

function	**base mgtde case**	$\|c\|$	**candidate mgtde case**
sec x	$\{1,1/2,5/24,61/720,277/8064\}$	1	$\{1,1/2,5/24,61/720,277/8064\}$
exp x^2	$\{1,1,1/2,1/6,1/24\}$	1	$\{1,1,1/2,1/6,1/24\}$
cosh x	$\{1,1/2,1/24,1/720,1/40320\}$	1	$\{1,1/2,1/24,1/720,1/40320\}$
tan x	$\{1,1/3,2/15,17/315,62/2835\}$	1	$\{1,1/3,2/15,17/315,62/2835\}$
arcsin x	$\{1,1/6,3/40,5/112,35/1152\}$	1	$\{1,1/6,3/40,5/112,35/1152\}$
sinh x	$\{1,1/6,1/120,1/5040,1/362880\}$	1	$\{1,1/6,1/120,1/5040,1/362880\}$
arctanh x	$\{1,1/3,1/5,1/7,1/9\}$	1	$\{1,1/3,1/5,1/7,1/9\}$

Evaluating the $d^{(5)}$ metric for each **candidate-target** pair, the best closed form approximation is given by

---

$$\textbf{target}_1 \simeq b_0 x^{-1/2} \sinh(x)$$

A similar argument may be applied to $\textbf{target}_2$.

7.5.2 Approximation when Closed Form Solution is Possible

We want the **closed form approximation** technique to recover the exact closed form solution if it exists. Below we present two examples to show that our algorithm is faithful. These examples are not meant to suggest that solution in series is the best way of solving these equations but rather that the **closed form approximation** algorithm converges to the exact closed form solution should one exist and its component parts be known to the library.

7.5.2.1 Closed Form Approximation by Composition

This problem is adapted from [Stephenson 78 p434]. The textbook solution and that presented below are somewhat different. Our approach uses the solution in series algorithm presented in *Chapter 6* whereas theirs relies on substituting derivatives of an abstract generalised power series into the equation, collecting similar terms, equating coefficients of linearly independent terms to zero and solving for the indicial indices and recurrence relations. This is a laborious process and would be costly to automate in a step for step fashion.

A second difference lies in the way the **target** series is recognised as being related to that of a "standard" function. The textbook approach does this "by inspection". We use the **closed form approximation** algorithm.

Together the solution in series and **closed form approximation** algorithms give us the leverage we require to find approximate solutions efficiently. Moreover, our technique converges to the exact closed form result should it exist and be constructible from the library entries.

Step (1) Solution in Series

Consider the differential equation

$$4x d^2y/dx^2 + 2dy/dx + y = 0$$

1i) put equation in normal form

$$d^2y/dx^2 + (1/x)(1/2)dy/dx + (1/x^2)((1/4)x)y = 0$$

1ii) verify conditions for solution in series satisfied

$q(x) = 1/2$ and $r(x) = (1/4)x$ are expandable as convergent power series at $x=0$.

1iii) compute expansions for $q(x)$ and $r(x)$

$q(x) = 1/2x^0 + 0x + 0x^2 + \ldots \equiv q_0 = 1/2 \wedge q_i = 0 \; \forall \, i>0$

$r(x) = 0x^0 + (1/4)x + 0x^2 + \ldots \equiv r_0 = 0 \wedge r_1 = 1/4 \wedge r_i = 0 \; \forall \, i>1$

1iv) evaluate $q(0)$ and $r(0)$, set up and solve indicial equation

$q(0) = q_0 = 1/2$

$r(0) = r_0 = 0$

indicial equation: $s(s-1) + q(0)s + r(0)$ becomes

$$s^2 - 1/2\, s = 0 \Rightarrow s = 0 \vee s = 1/2$$

1v) compute general form for recurrence relation

$$b_i = -(1/I(i+s)) \sum_{j=0}^{i-1} [(j+s)\, q_{i-j} + r_{i-j}]\, b_j, \quad i \geq 1$$

where $I(x) = x(x-1) + 1/2\, x = x(x-1/2)$

$$b_i = -(1/(i+s)(i+s-1/2)) \sum_{j=0}^{i-1} [(j+s)\, q_{i-j} + r_{i-j}]\, b_j, \quad i \geq 1$$

1vi) simplify recurrence relation

Consider the implications of the summation $\sum_{j=0}^{i-1}$. The smallest q_{i-j} in the sum is q_1. But $q_i = 0\ \forall\ i>0 \therefore$ the term in q_{i-j} makes no contribution. Conversely, the term in r_{i-j} does make a contribution as the smallest r_{i-j} in the sum is $r_1 = 1/4$. However, $r_i = 0\ \forall\ i>1$ so r_1. is the *only* non-zero term in the summation. $r_{i-j} = r_1$ iff $i-j=1$. Therefore, the recurrence relation simplifies to

$$b_i = -(1/(i+s)(i+s-1/2))\, r_1\, b_{i-1}, \quad i \geq 1$$

which, as $r_1 = 1/4$ becomes

$$b_i = -1 * (1/(4(i+s)(i+s-1/2)) * b_{i-1}, \quad i \geq 1$$

1vii) for each indicial index, identify corresponding selectors

As this is a one step recurrence relation the *selectors* for each value of the indicial index can only be $k^{(1/2)} = \{0\}$ and $k^{(0)} = \{0\}$

1viii) create recurrence structure

Collecting the above results together, the recurrence structure is

$\langle$*recurrence relation, selectors, indicial indices*$\rangle$ =

$$\langle b_i = -1 * (1/(4(i+s)(i+s-1/2)) * b_{i-1}\ \{k^{(1/2)}=\{0\}, k^{(0)}=\{0\}\}, \{s=1/2, s=0\}\rangle$$

Step (2) Mapping Recurrence Structure to Target Series

The *recurrence relation* implicitly defines two target series, one for $s=1/2$ and the other for $s=0$. As the sign of the recurrence relation is negative this means that signs of the coefficients will alternate. Taking the $s=0$ case first, $s=0$, $n=1$, $k=0$ and applying the mapping between recurrence structure and target series **target$_1$** becomes

---

target$_1$

indices: $\{j+0: j \text{ Qmod } 1 = 0 \wedge j \geq 0\}$

coeffs: $\{b_j: j \text{ Qmod } 1 = 0 \wedge j \geq 0\}$

signs: $\{+1: j \text{ Qmod } 2 = 0 \wedge j \geq 0\} \vee \{-1: j \text{ Qmod } 2 = 1 \wedge j \geq 0\}$

mgtde: $b_0 * \{1, 1/2, 1/24, 1/720, 1/40320\}$

Similarly, for the other case: $s=1/2$, $n=1$, $k=0$, **target$_2$** is

---

target$_2$

indices: $\{j+1/2: \text{j Qmod } 1 = 0 \wedge j \geq 0\}$

coeffs: $\{b_j: j \text{ Qmod } 1 = 0 \wedge j \geq 0\}$

signs: $\{+1: j \text{ Qmod } 2 = 0 \wedge j \geq 0\} \vee \{-1: j \text{ Qmod } 2 = 1 \wedge j \geq 0\}$

mgtde: $\{1, 1/6, 1/120, 1/5040, 1/362880\}$

Step (3) Finding a Set of CFAs to Target

3i) split target into as many series as there are k^s_j in $\{k^s\}$

in this case trivial as there is only one selector for each index

for each such series,

for each base series in library do

3ii) test whether base matches target

none of the bases in the library directly match **target$_1$**

3iii) and test whether $\exists\, c, m$: base $\circ\, cx^m$ = target

Condition is: $mp = n \wedge mq = s + k$ (theorem 7.1)
Instantiates to: $mp = 1 \wedge mq = 1/2$

succeeds for **target$_1$** with $n = 1$, $k = 0$, $s = 1/2$

Fix m

Theorem 7.1 is satisfied by the following base series with the corresponding value for p, q, m:

p	q	m	**functions**
2	0	$1/2$	$\cos x \quad \sec x \quad \cosh x \quad \mathrm{sech}\, x \quad \exp x^2$
1	0	1	$\exp x$

Fix sign c

The sign case of **target$_1$** is

signs: $\{+1 : j \bmod 2 = 0 \wedge j \geq 0\} \vee \{-1 : j \bmod 2 = 1 \wedge j \geq 0\}$

Filter the **base** series retaining those which, for some sign of c and under composition with cx^m, can be made signature equivalent to **target$_1$**

p	q	m	**functions**	**sign**(c)	**rule**
2	0	$^1/_2$	$\cos x \quad \operatorname{sech} x$	$c>0$	◦3
2	0	$^1/_2$	$\cos x \quad \operatorname{sech} x$	$c<0$	◦4a
1	0	1	$\exp x$	$c<0$	◦2

Each of these **base** series can therefore be made signature-index equivalent to **target**$_1$. However, in order to determine which is the best approximation (*i.e.* the **twin**) it is necessary to examine the relative magnitudes of the coefficients in the **candidate** and **target** series.

Fix magnitude of c

As the sign of the recurrence relation, σ, is –1, **rule** ◦**1** may be excluded from consideration as it will never yield a series whose signs alternate.

The magnitude case of **target**$_1$ is:

mgtde: $b_0*\{1,^1/_2,^1/_{24},^1/_{720},\ ^1/_{40320}\}$

Equate $a*\{|a_{i_1}c^{i_1}|, |a_{i_2}c^{i_2}|, ..., |a_{i_5}c^{i_5}|\}$ with $b_0*\{1,^1/_2,^1/_{24},^1/_{720},\ ^1/_{40320}\}$ and compute approximate values for $|c|$ and a from

$$|c| = [(|a_{i_1}| * |b_{j_2}|)/(|a_{i_2}| * |b_{j_1}|)]^{1/(i_2 - i_1)} \quad \wedge \quad a = b_0*(|b_{j_1}| + |b_{j_2}|)/(|a_{i_1}c^{i_1}| + |a_{i_2}c^{i_2}|)$$

Recall the notation: i_ℓ is the subscript of the ℓth non-zero coefficient in the **base** series and it's value is a_{i_ℓ}

p	q	m	**functions** $f(x)$	$\lvert c \rvert$	a	**CFA**
2	0	$^1/_2$	$\cos x \quad \operatorname{sech} x \qquad (c>0)$	1	b_0	$b_0 f(x^{1/2})$
2	0	$^1/_2$	$\cos x \quad \operatorname{sech} x \qquad (c<0)$	1	b_0	$b_0 f(-x^{1/2})$
1	0	1	$\exp x$	$^1/_2$	b_0	$b_0 f(^1/_2 x)$

Hence it is possible to evaluate the *magnitude cases* of the candidates

function	**base mgtde case**	$\|c\|$	**candidate mgtde case**
$\cos x$	$\{1,1/2,1/24,1/720,1/40320\}$	1	$\{1,1/2,1/24,1/720,1/40320\}$
$\operatorname{sech} x$	$\{1,1/2,5/25,61/720,277/8064\}$	1	$\{1,1/2,5/25,61/720,277/8064\}$
$\exp x$	$\{1,1,1/2,1/6,1/24\}$	1/2	$\{1,1/2,1/8,1/48,1/384\}$

The important point to realise is that each element, $|a_{i\ell}|$, of the *mgtde case* is multiplied by $|c^{i\ell}|$. However, this is only manifest in the exponential as $|c|$ happens to be unity in the other cases. In general this will not be so.

To find which of these is the best approximation, the $d^{(5)}$ metric is computed for each **candidate target** pair. Clearly, the metric is a minimum (in fact zero) for $\cos(x)$ suggesting that a good closed form approximation to $\mathbf{target_1}$ is

---

$$\mathbf{target_1} \simeq b_0 \cos(x^{1/2}) \vee b_0 \cos(-x^{1/2})$$

This is actually an exact solution which shows that the **closed form approximation** algorithm will find and indeed prefer the exact closed form solution if one exists and is known in the standard library. Notice, however, that the symmetry of cosine led to some duplication of effort. For the sake of efficiency, this suggests the **closed form approximation** algorithm needs to be amended to handle even or odd functions in a special way.

A similar analysis can be done for $\mathbf{target_2}$ which suggests the best closed form approximation is

---

$$\mathbf{target_2} \simeq \sin(x^{1/2})$$

Again the distance between **candidate** and **target** is zero suggesting $\sin(x^{1/2})$ is an exact closed form for the **target** series. Although truncated coefficient sequence equivalence does not guarantee strict equality between **candidate** and **target** it would require a very pathological case indeed to be untrue.

Pulling these two results together yields the general closed form approximation (in this case exact) to the solution of the original differential equation.

7.5.2.2 Closed Form Approximation by Multiplication

This problem is taken from [Boas 66 p558]. Again our method of solution differs from that of the textbook.

Step (1) Solution in Series

Consider the equation

$$x^2\, d^2y/dx^2 + 4x dy/dx + (x^2+2)y = 0$$

1i) put equation in normal form

$$d^2y/dx^2 + (1/x)(4)dy/dx + (1/x^2)(x^2+2)y = 0$$

1ii) verify conditions for solution in series satisfied

$q(x) = 4$ and $r(x) = x^2+2$ are expandable as convergent power series at $x=0$.

1iii) compute expansions for $q(x)$ and $r(x)$

$$q(x) = 4x^0 + 0x + 0x^2 + \ldots \equiv q_0 = 4 \wedge q_i = 0\ \forall\, i>0$$

$$r(x) = 2x^0 + 0x + 1x^2 + \ldots \equiv r_0 = 2 \wedge r_1 = 0 \wedge r_2 = 1 \wedge r_i = 0\ \forall\, i>2$$

1iv) evaluate $q(0)$ and $r(0)$, set up and solve indicial equation

$$q(0) = q_0 = 4$$

$$r(0) = r_0 = 2$$

indicial equation: $s(s-1) + q(0)s + r(0)$ becomes

$$s^2 + 3s + 2 = 0 \Rightarrow s = -1 \vee s = -2$$

Note that the indicial indices are different and differ by an integer.

1v) compute general form for recurrence relation

$$b_i = -(1/I(i+s)) \sum_{j=0}^{i-1} [(j+s)\, q_{i-j} + r_{i-j}]\, b_j, \quad i \geq 1$$

where $I(x) = x(x-1) + 4x + 2 = (x+1)(x+2)$

$$b_i = -(1/(i+s+1)(i+s+2)) \sum_{j=0}^{i-1} [(j+s)\, q_{i-j} + r_{i-j}]\, b_j, \quad i \geq 1$$

1vi) simplify recurrence relation

Consider the implications of the summation $\sum_{j=0}^{i-1}$. The smallest q_{i-j} in the sum is q_1. But $q_i = 0\ \forall\ i>0 \therefore$ the term in q_{i-j} makes no contribution. Conversely, the term in r_{i-j} does make a contribution as the smallest r_{i-j} in the sum is r_1 which precedes the non-zero term $r_2 = 1$. However, $r_i = 0\ \forall\ i>2$ so r_2. is the *only* non-zero term in the summation. $r_{i-j} = r_2$ iff $i-j=2$. Therefore, the recurrence relation simplifies to

$$b_i = -(1/(i+s+1)(i+s+2))\, r_2\, b_{i-2}, \quad i \geq 2$$

which, as $r_2 = 1$ becomes

$$b_i = -1 * (1/(i+s+1)(i+s+2)) * b_{i-2}, \quad i \geq 2$$

1vii) for each indicial index, identify corresponding selectors

Use the original unsimplified recurrence relation to evaluate for each *indicial index* s each $b_i\,(0 < i < 2)$. For this problem b_1 is the only coefficient to evaluate.

For $s = -1, i = 1$

$$b_1 = -(1/2)\,[(0-1)\,q_1 + r_1]\,b_0 = 0$$

For $s = -2, i = 1$

$$b_1 = -(1/0)\,[(0+1)\,q_1 + r_1]\,b_0 = (1/0)*0 \text{ which is indeterminate}$$

So the series for $s = -2$ will contain *two* arbitrary constants b_0 and b_1.

1viii) create recurrence structure

Collecting the above results together, the recurrence structure is

$\langle$*recurrence relation, selectors, indicial indices*$\rangle$ =

$$\langle b_i = -1 * (1/(i+s+1)(i+s+2)) * b_{i-2},\ \{k^{(-1)} = \{0\},\ k^{(-2)} = \{0, 1\}\},\ \{s = -1, s = -2\}\rangle$$

Step (2) Mapping Recurrence Structure to Target Series

The *recurrence relation* implicitly defines two target series, one for $s = -1$ and the other for $s = -2$. As the sign of the recurrence relation is negative this means that signs of the coefficients will alternate. Taking the $s = -1$ case first, $s = -1$, $n = 2$, $k = 0$ and applying the mapping between recurrence structure and target series $\mathbf{target_1}$ becomes

---

$\mathbf{target_1}$

indices: $\{j\text{-}1 : j \text{ Qmod } 2 = 0 \wedge j \geq 0\}$

coeffs: $\{b_j : j \text{ Qmod } 2 = 0 \wedge j \geq 0\}$

signs: $\{+1 : j \text{ Qmod } 4 = 0 \wedge j \geq 0\} \vee \{-1 : j \text{ Qmod } 4 = 2 \wedge j \geq 0\}$

mgtde: $b_0 * \{1, {}^1/_6, {}^1/_{120}, {}^1/_{5040}, {}^1/_{362880}\}$

Step (3) Finding a Set of CFAs to Target

3i) split target into as many series as there are k^s_j in $\{k^s\}$

in this case trivial as there is only one selector for each index

for each such series,

for each base series in library do

3ii) test whether base matches target

none of the bases in the library directly match **target$_1$**

3iii) and test whether $\exists\, c, m$: **base** $\circ\, cx^m =$ **target**

(this fails but we omit the details)

3iv) and test whether $\exists\, c, m$: **base** $*\, cx^m =$ **target**

Condition is: $p=n \wedge m=s+k-q$ (theorem 7.2)
Instantiates to: $p=2 \wedge m=-1-q$

succeeds for **target$_1$** with $n=2$, $k=0$, $s=-1$

Fix m

Theorem 7.2 is satisfied by the following **base** series with the corresponding value for p, q, m:

p	q	m	**functions**
2	0	−1	$\cos x$ $\sec x$ $\cosh x$ $\operatorname{sech} x$ $\exp x^2$
2	1	−2	$\sin x$ $\tan x$ $\arcsin x$ $\arctan x$ $\sinh x$ $\tanh x$ $\operatorname{arcsinh} x$ $\operatorname{arctanh} x$

Fix sign c

The sign case of $\mathbf{target_1}$ is

signs: $\{+1: j \bmod 4 = 0 \wedge j \geq 0\} \vee \{+1: j \bmod 4 = 2 \wedge j \geq 0\}$

Filter the **base** series retaining only those which for some sign of c and under multiplication with cx^m can be made signature equivalent to $\mathbf{target_1}$.

p	q	m	**functions**	**sign**(c)	**rule**
2	0	−1	$\cos x \quad \operatorname{sech} x$	$c > 0$	*3
2	1	−2	$\sin x \quad \arctan x \quad \tanh x \quad \operatorname{arcsinh} x$	$c > 0$	*3

Each of these **base** series can therefore be made *signature-index* equivalent to $\mathbf{target_1}$. However, in order to determine which is the best approximation (*i.e.* the **twin**) it is necessary to examine the relative magnitudes of the coefficients in the **candidate** and **target** series.

Fix Magnitude of c

The mgtde case of $\mathbf{target_1}$ is

mgtde: $b_0 * \{1, 1/6, 1/120, 1/5040, 1/362880\}$

Equate $a * \{|a_{i_1} c|, |a_{i_2} c|, \ldots, |a_{i_5} c|\}$ with $b_0 * \{1, 1/6, 1/120, 1/5040, 1/362880\}$ and compute approximate values for $|c|$ and a from

$$|c| = |b_{j_1}|/|a_{i_1}| \quad \wedge \quad a = b_0$$

where i_ℓ is the subscript of the ℓth non-zero coefficient in the **base** series and a_{i_ℓ} its value.

p	q	m	**functions** $f(x)$	$\lvert c \rvert$	a	**CFA**
2	0	−1	$\cos x$ $\operatorname{sech} x$	1	b_0	$b_0 x^{-1} f(x)$
2	1	−2	$\sin x$ $\arctan x$ $\tanh x$ $\operatorname{arcsinh} x$	1	b_0	$b_0 x^{-2} f(x)$

Hence it is possible to evaluate the *mgtde cases* of the candidates

function	**base mgtde case**	$\lvert c \rvert$	**candidate mgtde case**
$\cos x$	$\{1, 1/2, 1/24, 1/720, 1/40320\}$	1	$\{1, 1/2, 1/24, 1/720, 1/40320\}$
$\operatorname{sech} x$	$\{1, 1/2, 5/25, 61/720, 277/8064\}$	1	$\{1, 1/2, 5/25, 61/720, 277/8064\}$
$\sin x$	$\{1, 1/6, 1/120, 1/5040, 1/362880\}$	1	$\{1, 1/6, 1/120, 1/5040, 1/362880\}$
$\arctan x$	$\{1, 1/3, 1/5, 1/7, 1/9\}$	1	$\{1, 1/3, 1/5, 1/7, 1/9\}$
$\tanh x$	$\{1, 1/3, 2/15, 17/315, 62/2835\}$	1	$\{1, 1/3, 2/15, 17/315, 62/2835\}$
$\operatorname{arcsinh} x$	$\{1, 1/6, 3/40, 5/112, 35/1152\}$	1	$\{1, 1/6, 3/40, 5/112, 35/1152\}$

To determine the **twin** the $d^{(5)}$ metric is computed for each **candidate-target** pair. Clearly this is a minimum (zero) for sin x. Hence we conclude the best closed form approximation to $\mathbf{target}_1$ is

---

$$\mathbf{target}_1 \simeq b_0 (1/x^2) * \sin x$$

This is in fact an exact solution so again we see that the **closed form approximation** algorithm is faithful.

Finally the remaining case must be explored. Recall the recurrence relation was

$$\langle b_i = -1 * (1/(i+s+1)(i+s+2)) * b_{i-2}, \{k^{(-1)} = \{0\}, k^{(-2)} = \{0, 1\}\}, \{s = -1, s = -2\}\rangle$$

Now suppose $s = -2$, $k = 0$ or 1, $n = 2$. This gives rise to a second **target** series:

---

target$_2$

indices: $\{j\text{-}2: j \text{ Qmod } 2 = 0 \wedge j \geq 0\} \vee \{j\text{-}2: j \text{ Qmod } 2 = 1 \wedge j \geq 0\}$

coeffs: $\{d_j: j \text{ Qmod } 2 = 0 \wedge j \geq 0\} \vee \{e_j: j \text{ Qmod } 2 = 1 \wedge j \geq 0\}$

signs: $\{\text{sgn}(d_j) = +1: j \text{ Qmod } 4 = 0\} \vee \{\text{sgn}(d_j) = -1: j \text{ Qmod } 4 = 2\}$

$\{\text{sgn}(e_j) = +1: j \text{ Qmod } 2 = 1\} \vee \{\text{sgn}(e_j) = -1: j \text{ Qmod } 2 = 3\}$

mgtde: $\{d_0, e_1, {}^1/_2\, d_0, {}^1/_6\, e_1, {}^1/_{24}\, d_0, {}^1/_{120}\, e_1, {}^1/_{720}\, d_0, {}^1/_{5040}\, e_1, {}^1/_{40320}\, d_0, {}^1/_{362880}\, e_1\}$

Notice that the first ten elements of the *mgtde case* are computed because there are known to be two selectors for $s = -2$.

Step (3) Finding a Set of Closed Form Approximations to Target

3i) split target into as many series as there are k^s_j in $\{k^s\}$

In this case non-trivial as there two selectors for the indicial index $s = -2$. We break **target$_2$** into two independent series **target3** and **target4** and apply the closed form approximation algorithm on each series independently.

---

target$_3$

indices: $\{j\text{-}2: j \text{ Qmod } 2 = 0 \wedge j \geq 0\}$

coeffs: $\{d_j: j \text{ Qmod } 2 = 0 \wedge j \geq 0\}$

signs: $\{\text{sgn}(d_j) = +1: j \text{ Qmod } 4 = 0\} \vee \{\text{sgn}(d_j) = -1: j \text{ Qmod } 4 = 2\}$

mgtde: $d_0 * \{1, {}^1/_2, {}^1/_{24}, {}^1/_{720}, {}^1/_{40320}\}$

---

target$_4$

indices: $\{j\text{-}2 : j \text{ Qmod } 2 = 1 \wedge j \geq 0\}$

coeffs: $\{e_j : j \text{ Qmod } 2 = 1 \wedge j \geq 0\}$

signs: $\{\text{sgn}(e_j) = +1 : j \text{ Qmod } 2 = 1\} \vee \{\text{sgn}(e_j) = -1 : j \text{ Qmod } 2 = 3\}$

mgtde: $e_1 * \{1, 1/6, 1/120, 1/5040, 1/362880\}$

The details are entirely analogous to previous examples from this point onwards and we merely cite the result

$$\textbf{target}_2 = \textbf{target}_3 + \textbf{target}_4 \simeq d_0(1/x^2)\cos x + e_1(1/x^2)\sin x.$$

7.6 Limitations

The **closed form approximation** technique cannot accommodate equations with irregular singularities *e.g.* from [Boas 66 p575]

$$d^2y/dx^2 + [(1/x^2)/x^2]\,y = 0.$$

The problem is that $1/x^2$ cannot be expanded as a power series about $x = 0$. In such cases we detect that solution in series is inapplicable and terminate.

However, all is not lost. It is possible, as in this case, that **equation parsing** can determine a solution. We showed how this was done in *Chapter 5*.

A second way in which the **closed form approximation** technique can fail is through inadequacies in the library of **base** series. Clearly there is enormous latitude in choosing what to put in the library. The conflict is that the more functions we include, the slower the

processing. Our current choices are governed by what we found in mathematical handbooks [Spiegel 68] and expect engineers and physicists to know.

One way of incorporating more library entries lies in indexing them by properties of their **case** descriptions rather like a thesaurus. Whole groups of functions may then be eliminated at a stroke. Alternatively, in some applications certain classes of functions might be "expected" from previous work in the field. Is this is so, it might prove useful to bias the library with examples involving such functions.

So far we have included mainly "primitive" entries in the library *i.e.* functions such as $\sin(x)$ rather than compound ones such as $\sin(\cos(x))$. The value of including more sophisticated functions depends on how large a value of x one wants the closed form approximation to approximate the function. Unfortunately, the more complicated we make the library entries the harder they will be to understand qualitatively. So there is a balance between precision and comprehensibility.

Finally, the current rules for manipulating **base** series are limited to multiplication and composition with the simple polynomial cx^m. These could be extended by developing rules for more complicated polynomials. In addition there should also be rules for combining pairs of **base** series under various mathematical operations (*e.g.* see [Boas 66]).

7.7 Summary of Research Contributions of this Chapter

This chapter formalised the notion of **closed form approximation**. This is distinguished from **analytic abduction** and **equation parsing** by basing its approximations on the exact solution rather than reasoning via an abstraction of the original equation.

To do this we had to realize Sacks' desire to extend automated analysis to singular equations. The automation of solution in series was made tractable by utilising theorems from analysis which related the recurrence structure of the infinite series solution to

parameters in the original equation thereby circumventing a great deal of symbolic manipulation..

Mapping such a solution to a closed form approximation necessitated the invention of a novel representational scheme for infinite series which supported efficient computation. We defined a concept of distance between infinite series and two new notions of *signature index* and *coefficient sequence* equivalence. These allowed us to approximate, in closed form, functions which cannot be expressed exactly in closed form. We can confirm that such approximations faithfully preserve the local qualitative properties of the exact (infinite series) solution by numerical methods.

The technique could form the basis of a useful stand alone tool in computer aided engineering.

Chapter 8

Back Of The Envelope Reasoning

8.1 Introduction

This chapter describes **BOTHER**, a program for performing **B**ack-**O**f-**TH**e-**E**nvelope-**R**easoning. By this we mean the calculation of rough numerical bounds for expressions containing symbolic constants of unknown absolute value.

BOTHER works by exploiting knowledge of ordinal relations ($>, <, =, \geq, \leq$) to rewrite an expression in terms of a composite variable which is known to be "small". For example, given an expression, *expr*, containing symbolic constants X and Y and the ordinal relationships $0 < X < Y$ but no knowledge of the actual values of X and Y **BOTHER** will create the new variable X/Y which is bounded by $0 < X/Y < 1$ and attempt to rewrite *expr* in terms of X/Y. Having done this it becomes possible to use mathematical techniques to functionally approximate and subsequently bound the expression.

BOTHER implicitly makes the most conservative assumptions necessary about the relative magnitudes of X and Y sufficient to return a numerical estimate for *expr*. This policy induces an aggressive inference strategy, pruning away detail until a numerical estimate becomes possible.

The ability to simplify complex expressions down to rough approximations whilst holding rein just enough to keep some useful aspect left in the formulae is a powerful technique as it reduces the number of occurrences of an unknown and emphasizes the significant functional relationships.

These features reflect the key message of this thesis: that it is often better to sacrifice mathematical rigour for comprehensibility.

8.1.1 Plan

We begin by briefly reviewing the key features of **PRESS** as these provide the basis on top of which the magnitude reasoning capabilities of **BOTHER** are built. This is followed by a section detailing the kind of inference **BOTHER** can perform, the design of the program and the mathematical basis for the approximation technique.

The application of magnitude reasoning is essentially in two stages: a preprocessing step which rewrites an expression into a form where knowledge of relative magnitudes may be applied and the actual execution of the magnitude reasoning procedure. The second stage determines the exact nature (specifically the "order" explained below) of the approximation. We present the algorithm implementing the novel aspects of **BOTHER** and describe the two stages consecutively in §8.4.1 and §8.4.2 We follow this, in §8.5, with some concrete examples including those promised in *Chapter 3*. Finally we compare **BOTHER** with other systems to clarify how it differs from previous work. We conclude with some suggestions for extensions which could control the manner of approximation on the basis of meeting a desired accuracy rather than on expediency.

8.2 Overview of PRESS

The optimization stage of **analytic abduction** uses a cut down version of the equation solving system **PRESS** [Bundy & Welham 81, Bundy 83, Bundy & Silver 81, Bundy & Sterling 81, Sterling *et al.* 82]. The references describe the program in detail and we will merely highlight its key features so as to appreciate where magnitude reasoning fits in.

PRESS is a program which solves algebraic equations of the standard demanded by 'A' level mathematics courses. **PRESS** exploits the distinction between reasoning *about* an equation and reasoning *with* it. The former type of reasoning is called meta-level inference and the latter object-level inference. The key insight is that inference at the meta-level can be used to guide that at the object level.

The meta-level description of an equation is in terms of features such as the position or number of occurrences of a variable. This information is used to verify the preconditions for the invocation of various strategies for solving the equation. These include *isolation* (which solves equations containing a single occurrence of a variable by applying the inverse function to both sides), *collection* (which reduces the number of occurrences of an unknown in an equation thereby often enabling *isolation*) and *attraction* (which, as a preparatory step for *collection*, brings the occurrences of an unknown closer together).

PRESS has no notion about the relative magnitudes of the terms it manipulates. Consequently, if a subset of terms cannot be eliminated algebraically they will remain.

In practical applications, however, it is often useful to determine which of an equation's subterms are the most significant. Usually this is done by recognising order of magnitude relations between variables and propagating the magnitude relations through the functions in the equation. To do this often requires the equation to be *phrased* in a certain way. The new ingredient **BOTHER** brings to equation solving can be thought of as a *meta-magnitude level* by which we control the *phrasing* of an equation, the *degree* of approximation and the course of subsequent *simplification*.

Where appropriate, approximation can greatly reduce the complexity of an expression: in some cases, as we shall see, it permits numerical bounds to be placed on expressions containing symbolic constants of unknown absolute value.

8.3 Back-Of-the-Envelope-Reasoning

We describe the operation of **BOTHER** with respect to **analytic abduction** as this is the area in which it first arose. However, we emphasize that the ideas behind **BOTHER** could form the basis of a general purpose extension to **PRESS**.

Within **analytic abduction** **BOTHER** rules are initiated whenever the variable sought is an adjustable parameter from some trial function. However, a more general principle could conceivably be devised based on assigning a complexity measure to expressions and insisting that all expressions are within some complexity threshold. This, however, remains speculative at this stage.

8.3.1 The Origin of Ordinal Relations

The inequality assertions can arise from two sources:

- derived as a side effect of unifying a qualitative behaviour with a function descriptor
- derived from the mathematical solution of disjuncts

An example of the first kind will be seen in §8.5.1. Here we will show that a side effect of unifying the qualitative behaviour of a parachute, falling from initial velocity v_i, with the function descriptor for a falling exponential, is the (automatic) assertion that the newly discovered landmark value (representing terminal velocity) is bounded by $0 < \text{LMRK} < v_i$.

An example of the second process was seen in the bimolecular reaction, A+2B → C discussed in §3.5.4 and §3.6.3.1 of *Chapter 3*. The governing ordinary differential equation was

$$dn_c/dt = K(n_a - n_c)*(n_b - 2*n_c)^2$$

where n_a and n_b where the initial numbers of molecules of A and B respectively and n_c was the number of molecules of C formed after time t.

Qualitative simulation of this system suggested that n_c rose from zero to approach a constant value LMRK. **Analytic abduction** suggested a rising exponential as a possible interpretation of this behaviour and, consequently, that the approach was asymptotic. Therefore, $dn_c/dt \rightarrow 0$ as $t \rightarrow \infty$ and

$$K(n_a - n_c)*(n_b - 2*n_c)^2 = 0$$

which suggests $(n_a - n_c) = 0 \vee (n_b - 2*n_c) = 0$. Whichever member of the disjunct is satisfied first enforces an inequality between n_a and n_b.

8.3.2 Phrasing

BOTHER uses ordinal relations to construct a composite variable, called a *quotient variable*, which is provably of magnitude less than one, and then attempts to rewrite an expression in terms of the *quotient variable*. However, in general different sub-expressions will require different operations to convert their variables to *quotient variables*. For example, suppose $x<y$ and the expression is $(y - x)/(y+x)^{1.7}$. Then the *quotient variable* is x/y. In order to rewrite the numerator in the *quotient variable* requires division by y but to do so for the denominator requires division by $y^{1.7}$. **BOTHER** combines these operations into a single equivalence preserving map by recognising when one operation subsumes another.

8.3.3 Order of Approximation

The mathematical basis for the approximation method used in **BOTHER** lies in the ability to expand a continuously differentiable function as a power series in the vicinity of some point. As the re-phrased expression is written in terms of a *quotient variable*, x, which is known to

be of magnitude less than one, a convenient point about which to expand sub-expressions is $x=0$. Such an expansion is called a Maclaurin series and takes the form

$$f(x) = f(0) + f^{(1)}(0)*x + f^{(2)}(0)*x^2/2! + f^{(3)}(0)*x^3/3! + ... + f^{(n)}(0)*x^n/n! + R_n$$

where R_n is the remainder. $f(x)$ can be approximated by truncating this infinite series at some point. If the approximation includes all terms up to that in x^n we say $f(x)$ is approximated to order x^n and write this as $\mathbf{O}(x^n)$. For example, the Maclaurin series of $\cos(x)$ is

$$\cos(x) = 1 - x^2/2! + x^4/4! - x^6/6! + ... + (-1)^{n-1}x^{2n-1}/(2n-1)! + R_n$$

and successive approximations to increasing orders are as follows:

Order	Maclaurin Series	Error
$\mathbf{O}(x^0)$	1	$<x^2/2$
$\mathbf{O}(x^1)$	1	$<x^2/2$
$\mathbf{O}(x^2)$	$1 - x^2/2!$	$<x^4/24$
$\mathbf{O}(x^3)$	$1 - x^2/2!$	$<x^4/24$
$\mathbf{O}(x^4)$	$1 - x^2/2! + x^4/4!$	$<x^6/720$

Table 8-1. Successive Approximations of $\cos(x)$

Although it is possible to estimate the error in truncating the series at some term, **BOTHER** does not currently exploit this information but we suggest how a future version might use it in §8.9.

Note that the $\mathbf{O}(x^0)$ approximation is a constant but the higher order approximations ($\mathbf{O}(x^1)$, $\mathbf{O}(x^2)$, ...) contain the expansion variable explicitly. This provides a clean transition from algebraic approximation *via* the higher orders to numerical approximation as x tends to zero. Thus there is a natural continuum of functional approximation.

BOTHER begins by attempting to approximate expressions to the lowest order which explicitly includes the *quotient variable* (*e.g.* $\mathbf{O}(x^1)$ for $\sin(x)$ and $\mathbf{O}(x^2)$ for $\cos(x)$). Subsequently, if this effort fails, **BOTHER** reverts to computing a zero order approximation. This amounts to treating the ordinal relations as magnitude relations and is not, strictly speaking, sanctioned mathematically. In our application this abuse of ordinal relations is mitigated to some extent by recognising that "x" is always of magnitude less than one and hence positive integral powers of x become small rapidly. Moreover, real engineers do not appear to do error analysis either. Rather their strategy is to make the approximation, examine its ramifications and see if the answer leads to an obvious contradiction.

Such coercions between types of relation occur in other magnitude reasoning systems too *e.g.* **FOG** and the heuristic mode of **O[M]**† treat "slightly greater than" as a transitive relation which strictly speaking it is not. The justification for sanctioning these coercions is to permit more human-like inferences. And, for our goal of sacrificing accuracy in favour of comprehensibility they seem warranted. In fact, to accept complicated expressions without further (magnitude) simplification would give the impression of spurious accuracy.

Notice however that, for any $n>m$, the $\mathbf{O}(x^n)$ approximation *implicitly* makes a weaker assumption than the $\mathbf{O}(x^m)$ approximation. As **BOTHER** attempts the higher order approximation first it is at least conservative in making these "heuristic" inferences.

8.3.4 Task

The task **BOTHER** must perform is as follows. Given

- a set of input landmarks
- a set of discovered landmarks
- a mathematical expression involving landmark values
- a set of ordinal relations between the landmarks
- a goal parameter to be isolated

BOTHER is required to solve the expression for the goal which is free of discovered landmarks. This requirement induces an aggressive approximation strategy which often

† see [Raiman 86] and [Mavrovouniotis & Stephanopoulos 87]

requires subterms to be approximated numerically to eliminate their dependence on discovered landmarks. The rationale behind eliminating discovered landmarks is that the input to a model of a system is usually in terms of the accessible parameters (*i.e.* ones which can be measured) whereas discovered landmarks may not be accessible. Hence a simplification strategy oriented towards rewriting expressions in terms of known inputs is favoured.

8.3.5 Architecture

BOTHER is organised in the same way as **PRESS**. There are two levels: an *object level* which encodes rewrite rules and a *meta-level* which controls the application of these rules *via* the invocation of various equation solving methods. **BOTHER** rewrite rules differ from those of **PRESS** by having an extra argument representing the order to which the rewriting is performed. Hence a **PRESS** rule for *collection* such as

$$\sin(x)*\cos(x) \longmapsto \tfrac{1}{2}*\sin(2*x)$$

becomes the **BOTHER** rule

$$\sin(x)*\cos(x) \longmapsto x \mid_{\mathbf{O}(x^1)}$$

In future we can easily extend these rules to

$$\sin(x)*\cos(x) \longmapsto x \mid_{\mathbf{O}(x^2)}$$

$$\sin(x)*\cos(x) \longmapsto x - 2/_3x^3 \mid_{\mathbf{O}(x^3)} \textit{etc.}$$

As the original methods depend on such things as the number of occurrences of the unknown, the replacement of subterms in the unknown with numerical estimates has the effect of extending the original **PRESS** methods to cases where they do not immediately apply.

In addition to such augmented versions of *isolation*, *collection* and *attraction* **BOTHER** also has methods for *limit evaluation* and *bounding* which we discuss below.

8.4 Approximation in the Order Continuum

BOTHER has two distinct phases of operation: *preprocessing* which prepares an expression for approximation and *numerical estimation* which executes the approximation step, simplifies the resulting expression to an accuracy commensurate with that of the point of truncation and bounds the result.

8.4.1 Preprocessing

BOTHER retrieves the ordinal relations between landmarks, forms a *quotient variable*, q, out of their ratio, which is guaranteed to be of magnitude less than one, and rewrites the expression in terms of it. This requires finding an operation sufficient to rewrite each sub-expression in terms of the *quotient variable* and then combining these operations together. Moreover, the combined map must preserve equivalence with the original equation.

To do this **BOTHER** identifies a set of subterms called "*least dominating inclusive trees*" within the given expression. Each *least dominating inclusive tree* is the smallest expression tree (within a subterm) containing an instance of every variable mentioned in q. For example, if $q = x/(2*y)$ the set of *least dominating inclusive trees* of

$$goal = (y+\tfrac{1}{4}*x)^3/[(y-\tfrac{1}{2}*x)^{\frac{1}{2}} + y]$$

is

$$\{(y+\tfrac{1}{4}*x), (y-\tfrac{1}{2}*x)\}.$$

The trees are inspected to determine the operations required to rewrite each one in terms of the *quotient variable*. In the above example the suggested operations (shown in "⟨⟩") are

$$\{(y+\tfrac{1}{4}*x)\,\langle \div y\rangle, (y-\tfrac{1}{2}*x)\,\langle \div y\rangle\}$$

which yield, on factoring out the *quotient variable*,

$$\{(1+\tfrac{1}{2}*(x/(2*y))),\ (1-(x/(2*y)))\}$$

The problem now is that each *least dominating inclusive tree* is embedded in some outer function. To find the actual form of the required rewrite the suggested operations must therefore be rippled through these outer functions. For exponentiation and addition this is straightforward and the final suggested rewrites are

$$(y+\tfrac{1}{4}*x)^3 \qquad \longmapsto \qquad y^3*(1+\tfrac{1}{2}*(x/(2*y)))^3$$

and

$$[(y-\tfrac{1}{2}*x)^{\frac{1}{2}}+y] \qquad \longmapsto \qquad y^{\frac{1}{2}}*[(1-(x/(2*y)))^{\frac{1}{2}}+y^{\frac{1}{2}}].$$

In general, the *goal* will contain many sub-expressions each of which will give rise to different *least dominating inclusive trees* embedded in different functions. Consequently, the suggested operations they propose need to be merged to make some composite proposition. At the moment we can only do this if one operation subsumes the others. In the present case, as the final proposed operation on the numerator is $\langle \div y^3 \rangle$ and that on the denominator is $\langle \div y^{\frac{1}{2}} \rangle$ the combined operation becomes $\langle \div y^3 \rangle$ (similarly in §8.5.1).

We can summarise the preprocessing procedure in the following algorithm:

Algorithm 8-1: Preprocess Prior to Approximation

Input

- an equation of the form *goal* = *expression* and
- input landmarks
- discovered landmarks
- a set of ordinal relations amongst the parameters in *expression*.

Method

- find a quotient, q, of a subset of the parameters mentioned in *expression* which is provably < 1.
- parse *expression* finding all subterms which are *least dominating inclusive trees* of q.
- for each such tree, determine the operation needed to rewrite it, taking account of outer functions, into a form involving q
- combine the suggested operations into one composite operation
- apply this to *expression* to yield *newexpr*

8.4.2 Numerical Estimation

Having re-phrased the expression in terms of the *quotient variable* **BOTHER** then executes the approximation step. There are two methods for doing this; the second one being called only if the first one fails:

1) approximate *expr* to $\mathbf{O}(x^n)$ where n is the lowest order still containing x explicitly, simplify to $\mathbf{O}(x^n)$ and bound the result.

Otherwise,

2) approximate *expr* to $\mathbf{O}(x^0)$ (by computing $\lim_{(x \to 0)} expr$)

The first method implicitly imposes weaker assumptions about the relative magnitudes of the numerator and denominator than the second. *Limit evaluation* yields a numerical estimate directly without further bounding. It is important to realize that one cannot simply substitute zero for x to compute the $\mathbf{O}(x^0)$ approximation as it is possible that indeterminate forms could be generated *e.g.* 0/0. However, provided indeterminate forms are not generated direct substitution is permissible (and is in fact always the first *limit evaluation* technique to be tried).

8.4.2.1 Approximation by Truncation and Numerical Estimation of Bounds

Truncate

The choice of initial order of approximation is governed by the form of the Maclaurin expansion as it is the lowest order still retaining x explicitly. For example, if the sub-expression were $\cos(x)$ the initial order of approximation would be $\mathbf{O}(x^2)$, whereas if it were $\sin(x)$ it would be $\mathbf{O}(x)$. It is easy to pick this directly from the representation of Maclaurin expansions.

Simplification to $\mathbf{O}(x^n)$

If different sub-expressions are approximated to different orders then the whole expression is simplified to the *lowest* order. Hence, if an expression contained $\cos(x)/(1-x)$ the $\cos(x)$ is approximated initially to $\mathbf{O}(x^2)$, the $1/(1-x)$ to $\mathbf{O}(x)$ and hence the whole expression is simplified to $\mathbf{O}(x)$ prior to bounding.

Bound

Having obtained a uniformly simplified approximation, **BOTHER** then attempts to bound terms by exploiting the local monotonicity on (0,1) or (–1,0). For example $\log_e(1+x)$ is monotonic increasing (M^+) on (0,1) and is therefore bounded by $\log_e(1) < \log_e(1+x) < \log_e(2)$. Similarly, $\cos(x)$ is monotonic decreasing (M^-) on (0,1) and therefore bounded by $\cos(1) < \cos(x) < \cos(0)$. We encode such knowledge for each function for which we have a Maclaurin series.

This is a simple way of computing numeric bounds but we can sometimes tighten one of them if the variable in which the Maclaurin series is expanded can be factored into a numeric multiplier and algebraic multiplicand.

For example, let $x=n*y$ such that $0<x<1$, $0<n<1$ and $0<y<1$ and suppose the function for which an approximation is sought is $\log_e(1+x)$. The $\mathbf{O}(x^1)$ approximation is

$$\log_e(1+x) \longmapsto x\,|_{\mathbf{O}(x^1)}$$

However, as $x=n*y$ **BOTHER** can reason

$$\texttt{rule:}\quad 0<x<1 \wedge 0<y<1 \wedge 0<n<1 \wedge x=n*y \longmapsto 0<x<n$$

By using the monotonicity constraint the least upper bound on $\log_e(1+x)$ was $\log_e(2)$. Now it can be tightened to be n (assuming the $\mathbf{O}(y^1)$ approximation is adequate) which is tighter as $n<\log_e(2)$.

By using these strategies together **BOTHER** can find bounds for expressions in a *quotient variable*.

If a single value for an expression cannot be found immediately **BOTHER** obtains a lower and upper bounds on it. If the bounds are close enough (*i.e.* within a predetermined percentage difference) then **BOTHER** will return a compromise value.

Our philosophy is to choose an "order" of approximation such that the weakest magnitude assumption, sufficient to compute a numerical estimate, is the one selected.

Sometimes the truncation technique will fail. For example it will not always be possible to find a Maclaurin expansion for a sub-expression. In such cases **BOTHER** converts to the second estimation technique of taking the limit as the *quotient variable* tends to zero. This is more aggressive in the sense that it implicitly assumes a greater relative difference in magnitudes of the numerator and denominator of the *quotient variable*.

8.4.2.2 Approximation by Limit Evaluation

Limit evaluation subsumes direct substitution (which in fact is always the first method to be tried). If direct evaluation fails other methods are invoked depending on the form of the input. In the case of an indeterminate form (0^0, $0*\infty$, ∞^0, $\infty - \infty$, 0/0 or ∞/∞) being encountered, the form of the indeterminacy is used to guide the rewriting of the expression so that l'Hôpital's method becomes applicable.

Below we list the set of tactics we found to be useful in calculating some 50 example limits. These provide the basis for the set of rewrite rules we use to guide the calculation procedure. There are five commonly used methods: **direct evaluation, behaviour of logarithm, completion of the square, factorisation,** and **l'Hôpital's method**. Each tactic has associated preconditions which test its applicability.

Direct Evaluation

Preconditions:	nil
Method:	$\exists$ *newexpr* substitute(x_0,*x*,*expr*,*newexpr*) $\wedge$ simplify(*newexpr*,*lim*)
Postconditions:	$\neg$ indeterminate(*lim*)

Behaviour of logarithm

Preconditions:	matches($expr, f^g$)
Method:	$\exists$ *loglim* eval-limit($log_e(expr)$, *loglim*) $\wedge$
	simplify(e^{loglim} ,*lim*)
Postconditions:	$\neg$ indeterminate(*lim*)

Completion-of-the-square

Preconditions:	[matches($expr, (f+g)/h$) $\vee$
	matches($expr, (f-g)/h$) $\vee$
	matches($expr, h/(f+g)$) $\vee$
	matches($expr, h/(f-g)$)] $\wedge$ matches($f^2 - g^2, h$)
Method:	complete-sq(*expr, newexpr*) $\wedge$
	eval-limit(*newexpr, lim*)
Postconditions:	$\neg$ indeterminate(*lim*)

Factorisation

Preconditions:	factorizable(*expr*)
Method:	$\exists$ *factors* factorise(*expr, factors*) $\wedge$
	eval-limit(*factors, lim*)
Postconditions:	$\neg$ indeterminate(*lim*)

l'Hôpital's method

Preconditions:	direct-evaluation(*expr, lim*) $\wedge$ indeterminate(*lim*)
Method:	$\exists$ *quotient* indeterminate-form(*lim, indet*) $\wedge$
	rewrite(*expr, indet, quotient*)$\wedge$
	differentiate-numerator(*quotient,num*)$\wedge$
	differentiate-denominator(*quotient, denom*) $\wedge$
	eval-limit(*num/denom, lim*)
Postconditions:	$\neg$ indeterminate(*lim*) $\wedge$ $\neg$ derivative(*denom*, 0)

(*i.e.* the derivative of the denominator must not be zero).

Direct evaluation is the simplest method and ultimately all others terminate in a call to it. The system also knows of basic trigonometric rewrite rules which can be tried within the direct evaluation procedure.

The success of l'Hôpital's method rests in part on our ability to use the structure of an indeterminate form to guide the rewriting of the expression to be evaluated. For example, if the expression were of the form f^g and the indeterminate form were ∞^0 then a suitable rewrite would be

$$f^g = e^{(\log_e f)/(1/g)}$$

as, eventually, this would unfold to a subgoal of the form ∞/∞ as required for the correct application of l'Hôpital's method. The system currently knows of 9 such rewrite rules which can accommodate the relevant cases of the set of 50 limit calculations from which the above methods were abstracted and, in fact, constitute a more general set than required by these examples.

8.4.2.3 Algorithm

The approximation algorithm used by **BOTHER** can therefore be summarized as follows:

Algorithm 8-2. approximate expression

Input

- an expression, *expr*, written in terms of a *quotient variable*, x.

Method

- initialise order to be lowest order to include x explicitly $\mathbf{O}(x^n)$

(*) • if *order* = $\mathbf{O}(x^0)$

  - evaluate $\lim_{(x \to 0)} expr = expr \mid_{\mathbf{O}(x^0)}$

- else if *order* = $\mathbf{O}(x^n)$ $(n>0)$

  - expand subexpressions in x as Maclaurin series truncated at $\mathbf{O}(x^n)$ to yield *newexpr*, simplify *newexpr* to $\mathbf{O}(x^n)$ and return $expr \mid_{\mathbf{O}(x^n)}$
  - else decrement n and recurse from (*)

We have written the algorithm in a style suggestive of the change which would be necessary to extend the technique. Namely, by initialising the order to be $\mathbf{O}(x^2)$, $\mathbf{O}(x^3)$, $\mathbf{O}(x^4)$ *etc.* However, the present system is limited to low order approximations *i.e.* $\mathbf{O}(x^0)$, $\mathbf{O}(x^1)$ and $\mathbf{O}(x^2)$.

8.5 Examples

8.5.1 Approximation to $\mathbf{O}(x^0)$

Our first example was originally posed as an unsolved end of chapter problem in Acton & Squire [Acton & Squire 85 p47]. It concerns the motion of a parachute falling through the air subject to drag proportional to a power of its instantaneous velocity. The task is to relate the timescale of descent to the mass (m), resistance (b) and gravitational acceleration (g).

The equation of motion is

$$m*dv/dt + b*v^{1.7} = m*g$$

which is mapped to the constraint set

```
{deriv(v,f1), mult(m,f1,f2), M+(f3,v), mult(b,f3,f4), mult(m,g,f5),
                        add(f2,f4,f5)}
```

for simulation by **QSIM**. At the instant the parachute is opened, we assume the body is falling with a velocity v_0. So the quantity space for the velocity parameter is $\{-\infty, 0, v_0, +\infty\}$. Envisionment will predict that the body will be retarded in its descent and will eventually attain the newly discovered landmark value **LMRK** which lies somewhere in the interval $(0, v_0)$. Qualitatively the behaviour is as shown.

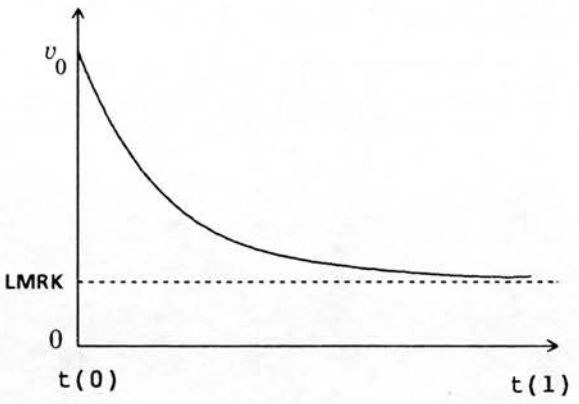


Figure 8-1. Qualitative Behaviour of a Falling Object in a Resistive Medium

In terms of the output from the envisionment this qualitative behaviour would be represented as a sequence of qualitative states at and between distinguished time points:

time	v	dv/dt
t(0)	<v_0,dec>	<(-∞ ,0),inc>
t(0,1)	<(0,v_0),dec>	<(-∞ ,0),inc>
t(1)	<LMRK,std>	<0,std>

which is revised to

time	v	dv/dt
t(0)	<v_0,dec>	<(-∞ ,0),inc>
t(0,1)	<(LMRK,v_0),dec>	<(-∞ ,0),inc>
t(1)	<LMRK,std>	<0,std>

by propagating knowledge of new landmarks back to the beginning of the envisionment.

Next a possible interpretation for this behaviour is obtained by finding a known analytic function whose function descriptor can be unified with the qualitative behaviour predicted by envisionment. Such a function is said to be *congruent*. For example the function descriptor of a falling exponential, $F(t)$, defined by

$$F = X + (Y - X)*\exp(-t/\tau)$$

is congruent to the above behaviour as its descriptor is

time	F	dF/dt
t(0)	<Y,dec>	<(-∞ ,0),inc>
t(0,1)	<(X,Y),dec>	<(-∞ ,0),inc>
t(1)	<X,std>	<0,std>

Hence, one possible interpretation of the parachute motion is that of a falling exponential in which LMRK is approached asymptotically (and hence t(1) is infinitely distant). Formally, our conjecture is then

$$v = \text{LMRK} + (v_0 - \text{LMRK})*\exp(-t/\tau)$$

where τ determines the natural time scale for the change. Under the interpretation that the behaviour is a falling exponential, the rate of change of velocity at t(1) becomes zero *i.e.* $dv/dt = 0$ at t(1) (infinite time). Hence,

$$b\,\text{LMRK}^{1.7} = mg$$

In order to obtain an approximate solution for $v(t)$ it is therefore necessary to express τ in terms of the other parameters in the model and the collocation fraction. Thus **BOTHER** substitutes the function conjecture into the differential equation and isolates τ.

$$\tau = [m(v_0 - \text{LMRK})c] / [\, b(\text{LMRK} + (v_0 - \text{LMRK})c)^{1.7} - mg]$$

Although this is formally correct, it is difficult to imagine how perturbations to parameters in τ affect its value. **BOTHER** makes this clearer by exploiting the ordinal relation $0 <$ LMRK $< v_0$ asserted during the matching of the qualitative behaviour to a function descriptor to create the *quotient variable* $\text{LMRK}/v_0 = q$ which is guaranteed to be bounded by $0 < q < 1$.

Stage 1: Preprocessing

Having identified a suitable *quotient variable*, **BOTHER** examines the subterms of the expression to determine the operation(s) required to rewrite each one in terms of q. In

general different subterms require different operations so **BOTHER** must find a composite operation. In the particular case at hand, the numerator requires division by v_0 whilst the denominator requires division by $v_0^{1.7}$. The composite operation is chosen to be division by $v_0^{1.7}$ as this subsumes division by v_0 and the expression is rewritten to

$$\tau = [m(1-q)\, c\, v_0^{-0.7}\, b^{-1}] / [(q + (1-q)c)^{1.7} - q^{1.7}]$$

Stage 2: Numerical Estimation

Approximation to $\mathbf{O}(x^1)$ is blocked because **BOTHER** cannot rewrite $(q + (1-q)c)^{1.7}$ into the form $(1+x)^n$ where $|x|<1$. Therefore **BOTHER** seeks an order $\mathbf{O}(x^0)$ approximation for τ by computing the limit as the *quotient variable* tends to zero. In this case a numerical estimate is obtained immediately without further bounding.

As no indeterminate forms are produced, direct substitution is permissible and the order $\mathbf{O}(x^0)$ approximation for τ is

$$\tau(q) \mid_{\mathbf{O}(x^0)} = (1/c^{0.7}) * (m/b{v_0}^{0.7})$$

The final form is much simpler to comprehend and gives a quick insight into how variations in system parameters are likely to affect the timescale in which the parachute slows the object down to a safe terminal velocity. This is of practical significance in estimating the height by which the parachute *must* be open to guarantee a safe landing.

8.5.2 Approximation to $\mathbf{O}(x^1)$

Our second example will concern approximation to order $\mathbf{O}(x^1)$. This requires a different bounding strategy from the order $\mathbf{O}(x^0)$ discussed above. In the more crude $\mathbf{O}(x^0)$ approximation *limit evaluation* is used to return a numerical value directly. In $\mathbf{O}(x^1)$ approximations an expression is first rewritten to an approximate functional form and *then*

bounded. Any extension of **BOTHER** to higher order approximations would have to adopt this strategy too.

The problem we consider was first encountered in *Chapter 3* §3.5.4 and concerns the chemical reaction A + 2B → C.

Stage 1: Preprocessing

The differential equation governing this process was

$$dn_c/dt = K*(n_a - n_c)*(n_b - 2*n_c)^2.$$

One of the function descriptors matching this behaviour was the rising exponential. Under this interpretation the final number of molecules of product C was approached asymptotically as time tended to `t(1)` (infinity). At `t(1)` the rate of production of C is zero so

$$0 = K*(n_a - n_c)*(n_b - 2*n_c)^2$$

which can be satisfied by either bracket becoming zero. Arbitrarily choosing the right hand bracket to become zero first imposes the ordinal relation $0 < n_b < 2n_a$. Consequently, a suitable *quotient variable* is $n_b/2n_a$.

Each trial function contained one or more adjustable parameters. The role of **BOTHER** within **analytic abduction** is to isolate those parameters. Using **PRESS** methods alone would often result in, at best, unwieldy expressions and possibly an inability to isolate a parameter. Consequently, **BOTHER** techniques are used to find approximate solutions.

Having rewritten expressions in the *quotient variable*, **BOTHER** was faced with the problem of finding an approximate form for β from

$$(1 - 1/\beta)\ \log_e(\log_e(c_1)/\log_e(c_2)) = \log_e(c_1/c_2) + \log_e(\ (1-(n_b/2n_a)(1-c_1))/(1-(n_b/2n_a)(1-c_2))\ )$$

given that $0 < n_b < 2n_a$, $0 < c_1 < 1$ and $0 < c_2 < c_1$. The techniques of **PRESS** can easily manipulate the terms such that β is isolated but the lack of knowledge of the absolute values of n_a and n_b prevent direct numerical evaluation.

BOTHER's first task is to identify where the trouble lies. It does this by inspecting the types of the subterms. As c_1 and c_2 have a known numerical value, **BOTHER** discovers that the difficult term is the double logarithm as all the others are free of the unknowns. Hence the focus shifts to

$$\log_e((1-(n_b/2n_a)(1-c_1))/(1-(n_b/2n_a)(1-c_2)))$$

Stage 2: Numerical Estimation

BOTHER has an inside-out simplification strategy. This means it starts with the most deeply nested structures and simplifies them before simplifying others. From the ordinal relations above **BOTHER** deduces $0 < n_b/2n_a < 1$, $0 < 1-c_1 < 1$ and $0 < 1-c_2 < 1$ and hence $0 < (n_b/2n_a)(1-c_1) < 1$ (similarly for the denominator). The key point is that although the absolute values of n_a and n_b are unknown their ratio is bounded and the bounds are computable. The strategy underlying **BOTHER** is to rewrite the expression in terms of this quotient and thereby bound the entire expression. Moreover, if the bounds are sufficiently tight, to pick a rough intermediate value. This is a simple idea but provides considerable leverage in terms of symbolic simplification.

Truncation

The $\mathbf{O}(x^1)$ approximation of $1/(1-x)$ where $x=(n_b/2n_a)(1-c_2)$ matches $(1+X)^N$ from the library of standard Maclaurin functions with $X = -x$ and $N = -1$. This sets up the preconditions for a rewrite rule

$$\texttt{expansion:}\quad |x|<1 \wedge y=1/(1-x) \longmapsto y=(1+x)\,|_{\mathbf{O}(x^1)}$$

and the substitution is made.

Simplification to $\mathbf{O}(x^1)$

BOTHER now detects that a further simplification is possible by the rule

$$\texttt{rule:}\quad |x|<1 \wedge |y|<1 \wedge z=(1+x)*(1+y) \longmapsto z=(1+x+y)\,|_{\mathbf{O}(x^1),\,\mathbf{O}(y^1)}$$

which together reduce the troublesome logarithm to $\log_e(1+(c_1-c_2)*(n_b/2n_a))$. Notice that this is an order dependent simplification and the cross term in "$x*y$" is omitted.

Bounding

At this point traditional simplifiers would be stumped as no further evaluation of $\log_e(1+(c_1-c_2)*(n_b/2n_a))$ seems possible without knowledge of the values of n_a and n_b. **BOTHER**, however, presses on. When confronted with an impasse, **BOTHER**, attempts to bound the expression.

In the particular case at hand, **BOTHER** retrieves the $\mathbf{O}(x^1)$ expansion of $\log_e(1+X)$, computes the lower bound from monotonicity considerations and the upper bound by recognising that $(c_1-c_2)*(n_b/2n_a)$ can be factored with $0 < n_b/2n_a < 1$. Hence the expression is bounded by

$$0 < \log_e(1+(c_1-c_2)*(n_b/2n_a)) < (c_1-c_2).$$

In general, if this is successful and the computed bounds are close enough, **BOTHER** suggests an approximate value. This is a very powerful inference as it permits a significant reduction in the complexity of an expression.

Hence the expression for β becomes

$$\beta = [\log_e(\log_e(c_1)/\log_e(c_2))] / [\log_e(\log_e(c_1)/\log_e(c_2)) - \log_e(c_1/c_2) - \log_e(1+(c_1-c_2)*(n_b/2n_a))]$$

which, with $c_1 = 2/3$ and $c_2 = 1/5$, is bounded by

$$0.452 < \beta < 0.534.$$

BOTHER therefore suggests an approximate value of $\beta = 0.49$.

The enormous reduction in complexity that **BOTHER** inferences impart considerably enhances the comprehensibility of the approximate solutions proposed by **analytic abduction**.

8.6 Library of Standard Maclaurin Series

We adopted a crude representation of explicit expansions of "standard" Maclaurin series. A more elegant approach would have allowed **BOTHER** to compute Maclaurin expansions of unfamiliar functions from first principles. Although our symbolic manipulator could certainly handle the differentiations required to *formally* obtain a series, we shied away from this because we should still have to prove that it converged to the original function and the computational overhead in doing this would have been high.

We therefore restricted ourselves to a pre-determined set of function-in-limit expansions for which these conditions are satisfied. This is not that unreasonable as it seems unlikely that real engineers generate these series from first principles each time they are employed. Rather we expect them to be part of their core mathematical knowledge.

Our philosophy, then, is to approximate a function *via* its truncated Maclaurin series. As we are only ever concerned with expansions in *quotient variables* (of magnitude less than one) convergence is guaranteed.

We have chosen the set of "standard" series to be those appearing in a respected handbook of mathematical formulae [Spiegel 68]. Letting x represent the *quotient variable*, these include series for $(1+x)^n$, $\exp(x)$, $\log_e(1+x)$, $\frac{1}{2}\log_e((1+x)/(1-x))$, $\sin(x)$, $\cos(x)$, $\tan(x)$, $\cot(x)$ $(x \neq 0)$, $\sec(x)$, $\csc(x)$ $(x \neq 0)$, $\arcsin(x)$, $\arccos(x)$, $\arctan(x)$, $\operatorname{arccot}(x)$, $\sinh(x)$, $\cosh(x)$, $\tanh(x)$, $\coth(x)$ $(x \neq 0)$, $\operatorname{sech}(x)$, $\operatorname{csch}(x)$ $(x \neq 0)$, $\operatorname{arcsinh}(x)$, $\operatorname{arctanh}(x)$. In all cases we give the series that is convergent for $0<|x|<1$. The case $x=0$ corresponds to an $\mathbf{O}(x^0)$ approximation and this would be dealt with by limit evaluation.

8.7 Comparison with other Work

We have already acknowledged the influence of **PRESS** on the design of **BOTHER**. However, **PRESS** concentrated on the control of equation solving without regard for the relative magnitudes of the terms being manipulated. In order to place **BOTHER** in proper perspective

we must also clarify how it differs from previous Artificial Intelligence systems that have dealt with magnitude reasoning.

There are three dimensions we can use to compare previous systems:

- the type of *relations* they used
- the type of *reasoning* permitted over those relations
- the kind of *task* performed by each system

The table below compares **BOTHER** with a related systems along these dimensions. A bullet indicates the corresponding system has the adjacent *relation*, *reasoning* capability or *task*.

	BOTHER	FOG	O[M]	BOUNDER	QUANT. LATTICE	BENNETT	CHEPA.
Ordinal Relations	•			•	•		
Magnitude Relations		•	•			•	
Numerical Reasoning	•				•	•	
Interval Reasoning			•	•	•		•
Qualitative Reasoning		•					•
Math'l Reasoning	•			•	•	•	
Error Estimation						•	
Equation Solving	•	•	•			•	
Bounding	•			•	•		
Simulation							•
Functional Approx'n	•		•				

Table 8-2. Comparison of Relations, Reasoning Style and Task

8.7.1 Relation Dimension

The early qualitative reasoning systems [de Kleer & Brown 84, Forbus 84, Kuipers 86a] placed the most significant emphasis on the *sign* of some quantity. Using only the signs of quantities can lead to ambiguities in qualitative arithmetic as something positive plus something negative could be positive, zero or negative.

To overcome this, it is necessary to incorporate magnitude information into the constraints. However, in these early systems magnitude information was at best merely ordinal which limited the power of the reasoner. For example, if a quantity space was defined by the landmark set $\{-\infty$, **LMRK1**, 0, **LMRK2**, **LMRK3**, $+\infty\}$ ordinal relations are sufficient to answer queries of the form "is **LMRK2** < **LMRK3**?" *etc* but can say nothing about the relative magnitudes of two landmarks if one is positive and the other negative. Nor can they say by *how much* **LMRK2** is less than **LMRK3**.

In many circumstances the ambiguity caused by the lack of absolute magnitude information resulted in unnecessary branching in an envisionment thereby obscuring the underlying behaviour of the system.

A useful criterion, then, for comparing different systems purporting to use magnitude information is whether or not they employ *ordinal* or *magnitude* relations.

BOUNDER [Sacks 87c] and the **QUANTITY LATTICE** [Simmons 86] are similar to **BOTHER** in that they use ordinal relations. **FOG** [Raiman 86], **O[M]** [Mavrovouniotis & Stephanopolous 87] and Bennett's system [Bennett 87] uses magnitude relations. The **CHEPACHET** system [Davis 87a] has neither magnitude nor ordinal relations explicitly but, because the qualitative values it manipulates are signed magnitude intervals, magnitude relations are implicit.

In a sense, **BOTHER** spans both camps by taking the ordinal relationship between two quantities, forming a *quotient variable* from them and implicitly making the weakest magnitude assertion between its numerator and denominator sufficient to compute a numerical bound. This is the inference governing the choice of order of approximation.

8.7.1.1 Magnitude Relations

Each of the magnitude reasoning systems mentioned above has its own set of relations.

FOG has 3 primitive magnitude relations:

$X \ll Y$ (X is negligible in relation to Y, and may be of either sign)

$X \simeq Y$ (X has the same sign and is close to Y *i.e.* $(X-Y) \ll Y$)

$X \sim Y$ (X has the same sign and order of magnitude as Y)

N.B. Raiman uses **Ne** ($\ll$), **Vo** ($\simeq$) and **Co** ($\sim$). The change is only intended to make the rules more readily comprehensible. The **FOG** system comprises 30 rewrite rules combining these relations with qualitative signs and standard addition and multiplication.

O[M] has 7 primitive magnitude relations between positive quantities and 21 compound relations defined as a span between two primitive relations.

$X \ll Y$ (X is much smaller than Y)

$X -< Y$ (X is moderately smaller than Y)

$X \sim< Y$ (X is slightly smaller than Y)

$X = = Y$ (X is exactly equal to Y)

$X >\sim Y$ (X is slightly larger than Y)

$X >- Y$ (X is moderately larger than Y)

$X \gg Y$ (X is much larger than Y)

One difference between **O[M]** and **FOG** is that **O[M]** relations are given a strict semantics based on the ratio of X/Y with respect to unity and a domain dependent accuracy parameter θ, whose value lies between 0 and 1. For example, $X/Y \sim< 1$ means that $X/Y \in [1/(1+\theta), 1)$ where θ is necessarily less than one and typically between 0.05 and 0.20 (5% to 20%). The scale of all **O[M]**'s magnitude relations ultimately is set by the choice of θ.

BOTHER similarly rephrases an ordinal relation, $X < Y$, to create a *quotient variable* $X/Y < 1$. However, whereas **O[M]** has an explicit handle on the relative magnitudes of X and Y **BOTHER** makes an implicit assumption sufficient to compute a numerical estimate.

Bennett uses two magnitude relations

$X \ll Y$ (defined informally as "negligible with respect to all other quantities in a formula")

and

$X \approx Y$ (approximately equal to)

8.7.1.2 Ordinal Relations

The **QUANTITY LATTICE** is a directed graph whose nodes are symbolic variables, numbers or simple arithmetic expressions (collectively termed *quantities*) and whose arcs testify to the ordinal relationship ($>$, $<$, $=$, $\geq$, $\leq$, or $\neq$) between the nodes they join.

The ordinal relations in **BOUNDER** and **BOTHER** are entirely standard except that they are over the extended reals permitting rewrites of the form $\log_e(0) = -\infty$.

8.7.2 Reasoning Dimension

A second dimension that can be used to compare different magnitude reasoning systems concerns the kinds of inferences each supports. We identify four types: numeric (dealing with numbers), interval (dealing with numeric or symbolic ranges), qualitative (dealing with the signs of quantities) and mathematical (dealing with functions).

8.7.2.1 Numeric Reasoning

Numeric reasoning in the **QUANTITY LATTICE** translates numerical values to ordinal relations *e.g.* given $X \leq 1$ and $Y = 2$ the **QUANTITY LATTICE** infers $X < Y$.

Bennett's system and **O[M]** have a similar facility for concluding magnitude relations. For example, given $X = 0.01$ and $Y = 1000$ each can infer $X \lll Y$ (although the meanings of "$\lll$" are not the same). **FOG** cannot draw such an inference. In addition, Bennett uses numerical reasoning to estimate the error of the approximations.

Numerical inference in **BOTHER** is used to pick a representative value for a bounded parameter if its bounds differ by less than a predetermined percentage. However, there is no mechanism for explicitly inferring a magnitude relationship between two variables given values for them.

8.7.2.2 Interval Reasoning

Both the **QUANTITY LATTICE** and **BOUNDER** use a form of *bounds propagation* for computing upper and lower bounds on an expression given numerical ranges for sub-expressions. Each bound is computed by combining the **sup** and **inf** over a domain in the most pessimistic manner *e.g.* for a sum $[3,6)+[-1,5]$ yields $[2,11)$.

BOUNDER, however, uses a second interval bounding method, *iterative approximation*, which performs *bounds propagation* over a sequence of sub-domains. Intuitively, the bigger the domain the more likely that it will contain the "worst" **sup** and **inf** *i.e.* the values that maximise the upper bound and minimize the lower bound. Consequently iterative approximation which deals with a sequence of smaller domains tends to find tighter bounds but at the expense of more work.

The **QUANTITY LATTICE** can also find ordinal relationships between quantities whose values are specified by intervals provided those intervals do not overlap.

CHEPACHET uses an interval calculus to reason about how ranges of a parameter change under operations in qualitative arithmetic. Each range is represented by a symbolic term (*e.g.* **SMALL**, **MEDIUM**, **LARGE**) or by a spanning sequence of such terms. Qualitative multiplication, for example, is defined *via* a table of interval mapping rules such as **LARGE** $\otimes$ **-MEDIUM** = **-LARGE**.

BOTHER performs no interval reasoning whatsoever.

A recent extension to interval reasoning [Mavrovouniotis 89] is to replace single continuous intervals with disjoint interval tuples of the form $\{(-x_1,-x_2) \cup (0,0) \cup (+x_3,+x_4)\}$. Here single value are represented as an interval with zero extent *e.g.* (0,0). In a traditional interval calculus, given $y=(1,1)/(-1,1)$ (*i.e.* $y=1/x$ where $x \in (-1,1)$) it would only be possible to conclude $y \in (-\infty, +\infty)$ [Mavrovouniotis 89 p34]. However, with the new syntax for intervals the bounds can be made more precise and limit y to $y \in \{(-\infty, -1) \cup \varnothing \cup (1, +\infty)\}$ with zero avoided (represented as null). When such inferences are chained together the idea is to collapse the negative and positive intervals into maximal ones to prevent combinatorial explosion of disjuncts. Consequently some information is lost but this does nevertheless provide a more useful set of distinctions than the older methods. Perhaps most usefully it can, in the appropriate circumstances, allow a system to conclude that some parameter is non-zero. Such inferences are required in order to establish preconditions for performing certain mathematical operations *e.g.* division.

Problems with interval calculi have been well documented by Davis [Davis 87b] & Struss [Struss 87, Struss 88a, Struss 88c].

8.7.2.3 Qualitative Reasoning

FOG has rewrite rules involving both magnitude relations and qualitative values (the parenthesis function "[]"returns the sign of its argument) *e.g.*

R_{17}: $[A]=[B], A \simeq C \rightarrow (A+B) \simeq (C+B)$

The relations mix both sign and magnitude information as can be seen from rules R_9 and R_{10} which state that

R_9: $A \simeq B \rightarrow [A]=[B]$

R_{10}: $A \sim B \rightarrow [A]=[B]$

whereas rule R_{11} states

R_{11}: $A \lll B \rightarrow -A \lll B$

The magnitude relations in **O[M]** are between strictly positive or zero quantities. Reasoning about signs is done elsewhere.

8.7.2.4 Mathematical Reasoning

Mathematical reasoning in the **QUANTITY LATTICE** uses a known relationship between two variables to compute the relationship between arithmetic combinations of them *e.g.* given $X \leq 1$ and $Y = 2$ the **QUANTITY LATTICE** can infer a relationship between $X + X$ and Y. Moreover, it can spot when common terms cancel either side of a relation *e.g.* given $A > C$ and $A = B + X$ and $C = D + X$ infer $B > D$.

BOUNDER is, mathematically, more powerful than the other systems. Its *substitution* technique can replace the bound on a variable with terms in other variables and then bound those. The advantage of this is that it implicitly enforces known ordinal relations. For example, if $X \leq Y$ then *substitution* bounds their difference above by zero *i.e.* $X - Y \leq 0$ by substituting Y for the upper bound of X and evaluating $X - Y \leq Y - Y = 0$. The previous *bounds propagation* technique can only restrict $X - Y \leq \infty$.

BOUNDER can also bound multivariate functions by determining the zeroes of their partial derivatives (checking maxima or minima from the signs of their second partial derivatives) and substituting as appropriate. This facility might prove useful in maximizing the formula for the error incurred in truncating a Maclaurin series after n terms. If this were possible we would be able to extend **BOTHER** with an error estimation capability.

Like **BOTHER**, **O[M]** can deduce functional approximations by truncated Maclaurin series *via* rules of the form

$$X \ll 1 \rightarrow \exp(X) > \sim 1 + X$$

and

$$X \ll 1 \rightarrow \sin(X) > \sim X$$

8.7.3 Task Dimension

Two basic kinds of task are possible: either the system constructs an approximate solution or else verifies a magnitude proposition.

8.7.3.1 Equation Solving

FOG, O[M], Bennett's system and BOTHER are all effectively approximate equation solvers. However, BOTHER and O[M] are the only ones which make *functional* approximations.

Unlike FOG, O[M] and Bennett, BOTHER does not manipulate given magnitude relations, it implicitly *creates* them. BOTHER's task is to aggressively rewrite complex expressions by finding rough numerical approximations for subterms. It is essentially exploratory, pushing an expression to an extreme. In our application this is precisely what we want as this often highlights the key functional relationships.

Of the systems discussed, Bennett's is the only one to estimate the error of the approximation.

Given a set of constraints involving arithmetic operators and magnitude relations FOG is required to find the qualitative value of some parameter. Direct rewriting is not always sufficient so FOG uses *reductio ad absurdum* (in the example cited in Raiman's paper a parameter was proven to be "+" by deducing contradictions if it were "0" or "–"). However, there is little guidance as to how the firing sequence of FOG rules is determined.

Rule invocation in BOTHER is directed by meta-level schemata similar to those in PRESS. BOTHER's goal is explicit and its rewrite effort therefore focused.

O[M]'s inferences are also goal directed. The user can either state that the goal is to relate two quantities or that alternative relations between two parameters should be explored. The same technique is used transparently: O[M] simply sets up seven assumptions corresponding to the seven primitive relations and attempts to prove each one. Hopefully only one is provable from the current knowledge. Depending on whether "strict" or "heuristic" interpretation semantics are used this is more or less likely.

Bennett's program combines quantitative and qualitative knowledge in reasoning about systems specified by sets of equations. The quantitative knowledge is in terms of negligibility ($\ll$) and approximate equality ($\approx$) relations between parameters and the inference over them is encoded as rules. The negligibility assertions arise automatically by executing rules encoding domain knowledge. For example, taking physical chemistry as his domain, Bennett cites the rule:

> *IF the acid is present at a much greater concentration than* 10^{-7} *molar,*
> *THEN* $[OH^-]$ *will be negligible compared to all other species.*

This is in contrast to **BOTHER** where ordinal relations are derived automatically from qualitative simulation or the solution of mathematical disjuncts.

In **BOTHER** we were interested in building a general purpose exploratory approximation system which was totally separate from any domain knowledge, emulated the behaviour of real engineers and had an explicit meta-magnitude level for controlling the *phrasing* of expression, the *degree of approximation* and the course of subsequent *simplification*.

8.7.3.2 Bounding

BOUNDER and the **QUANTITY LATTICE** were built to answer queries about ordinal relationships between expressions and to impose bounds on variables occurring in them.

If a node in the **QUANTITY LATTICE** is accessible by more than one route, and the different paths impose ordinal relations of differing specificity, the different relations are combined to find the tightest bound possible. For example, if it can be shown that quantities X and Y are related by $X \leq Y$ via one path and $X \geq Y$ via another then the relationship recorded in the lattice is $X = Y$.

BOUNDER also determines the ordinal relationship between two quantities by bounding their difference. In particular, it derives the upper and lower bounds on a variable, X, occurring in an inequality, $L \leq R$, by re-phrasing it as $X \leq U$ or $X \geq U$ with U free of X [Sacks 87c p650]. Similarly for equalities. The insistence that U be free of X is crucial. If X cannot

be isolated *e.g.* if the inequality were something like $X \leq 2^X$, **BOUNDER** *ignores* the constraint.

BOTHER however, is quite different: by approximating subexpressions with numeric values, a variable which appears to be algebraically caged in an uninvertible formula can be effectively isolated by replacing its troublesome occurrence(s) numerically. This is a powerful inference as it permits an approximate solution to be obtained for an equation that cannot be solved by purely algebraic means.

Moreover, the bounding technique in **BOTHER**, whilst being much simpler than the techniques in **BOUNDER**, is sufficient for our purposes given that we have contrived our *quotient variable* to be of magnitude less than one and the functions we deal with are monotonic on (0,1) (although possibly non-monotonic on a larger domain).

8.7.3.3 Simulation

The task of **CHEPACHET** was to perform a qualitative simulation which incorporated magnitude reasoning directly into the envisionment process. The algorithm used is similar in spirit to that driving **ENVISION** [de Kleer & Brown 84].

8.8 Limitations

BOTHER currently does not analyse the error in its approximations. In common with real engineers and most other magnitude reasoning systems it places the onus on the user to adjudge its adequacy. This is not ideal as we would like to be able to assign an accuracy to our approximations.

The mechanism for re-phrasing an expression in terms of the *quotient variable* has proved adequate so far but it is easy to conceive of examples where the rippling through of the operation past outer functions is impossible. Moreover, even if a set of operations can be

determined there is no guarantee that they can be merged successfully into one composite operation. However, for the cases we have considered, which involve only polynomials, the procedure is successful.

Finally, **BOTHER** cannot currently compute Maclaurin expansions for arbitrary functions. The differentiation ability required exists but the ability to check the convergence properties of the series it proposes does not. This is important because although we can formally calculate an expansion this alone does not guarantee that it converges to the real function.

8.9 Future Extensions

This chapter has concentrated on using ordinal magnitude information to rewrite expressions into an approximate form. An alternative which we have not explored is to use relative magnitude information to simplify the initial model. For example, the equation of motion of a pendulum, of length ℓ, with gravitational acceleration g is

$$d^2\theta/dt^2 + (g/\ell)*\sin(\theta) = 0$$

The $\sin(\theta)$ term can be approximated by a truncated Maclaurin series. Consequently, a sequence of approximate models could be devised as follows

Order	**Approximation**
$\mathbf{O}(x^1)$	$d^2\theta/dt^2 + (g/\ell)*\theta = 0$
$\mathbf{O}(x^3)$	$d^2\theta/dt^2 + (g/\ell)*(\theta - \theta^3/6) = 0$
$\mathbf{O}(x^5)$	$d^2\theta/dt^2 + (g/\ell)*(\theta - \theta^3/6 + \theta^5/120) = 0$

This strategy of approximating the *model* rather than approximating the *solution* runs the risk of suppressing important features. For example, the solution of the $\mathbf{O}(x^1)$ approximation of the pendulum equation suggests the period is independent of the

amplitude of oscillation. However, the solution of the $\mathbf{O}(x^3)$ (and above) approximations contains a dependence on the initial amplitude. Such observations suggest that model approximation is not as straight forward as it might initially appear.

An initial foray into this area has been made by Iwasaki & Bhandari [Iwasaki & Bhandari 88]. Their approach rests on the intuition that, when describing the behaviour of a complex dynamic system which has many variables, some of the variables are more tightly connected than others and it is possible to describe the short term behaviour of the whole system by the behaviour of each sub-system independently.

To turn this intuitive idea into an approximation technique, the authors proceed as follows. The dynamic system is modelled by a system of first order differential equations. Then, an approximate model is constructed by rewriting the matrix of coefficients in (almost) block diagonal form. This allows a reduction in the number of degrees of freedom in the system and leads to approximate equations for so called "aggregate" variables. The value of the approximation depends on how large the off-diagonal elements are (a completely decomposable system has zero elements off the diagonal blocks). So the technique works best for systems which are "almost decomposable".

A second extension to the technique developed in this Chapter, would be to incorporate dynamic error analysis. Formulae for the error incurred in truncating a Maclaurin series after a finite number of terms are well known. However, the most general version includes an existentially quantified parameter which makes it hard to evaluate. I suspect some numerical scheme exists to maximise this function and hence bound the error but as yet I have not been able to solve this problem.

If automatic error analysis *could* be performed I envisage using it to determine the optimum truncation point for some functional approximation. This would be an alternative to the aggressive but expedient strategy **BOTHER** currently uses.

8.10 Summary & Conclusions

8.10.1 Task, Architecture & Algorithm

BOTHER is an exploratory approximate equation solving system which uses knowledge of the ordinal relationship between symbolic constants of unknown absolute value to bound expressions containing them. The idea is to implicitly make the weakest assumption about the relative magnitudes of the unknowns sufficient to compute a numerical estimate for difficult subterms. This policy induces an aggessive inference strategy pruning away detail until a numerical estimate becomes possible.

The need for **BOTHER** arose out of the demand that the predictions of **analytic abduction** be readily comprehensible. If, on substituting a trial function into a differential equation, an unwieldy expression is produced the key functional relationships can be obscured. They can often be revealed, however, by considering an extreme case where some subterms are small (and may be rewritten into simpler forms) or negligible (and eliminated altogether). Engineers are particularly good at this kind of inference and **BOTHER** attempts to mimic their reasoning processes.

When confronted with an expression containing symbolic constants of unknown value **BOTHER** attempts to find a relative magnitude ordering between them. This determines a ratio of the two unknowns which is provably of magnitude less than unity. The expression is then rewritten in terms of this quotient (if possible) and then approximated as a truncated Maclaurin series.

BOTHER selects the initial order of approximation to be the lowest order truncation still containing the expansion (*i.e. quotient*) variable explicitly. If different sub-expressions are approximated to different orders **BOTHER** simplifies the whole expression to the lowest order. Having done this, sub-expressions in the *quotient variable* are bounded by applying local monotonicity. If the bounds are sufficiently close **BOTHER** takes the pragmatic view of picking a mid point "representative" value.

If an $\mathbf{O}(x^n)$ $(n>0)$ approximation is blocked, *e.g.* by being unable to find a Maclaurin expansion, *limit evaluation* is invoked which implicitly pushes the ordinal relations to an

extreme resulting in an $\mathbf{O}(x^0)$ approximation. In the latter case no further numerical bounding is necessary as limit evaluation returns a numerical result directly.

In forming the $\mathbf{O}(x^0)$ approximation, it is not always sufficient to substitute zero for the *quotient variable* as this can sometimes result in indeterminate forms *e.g.* 0/0 being generated. Hence *limit evaluation* uses a collection of progressively more sophisticated methods; direct evaluation being the first attempted.

8.10.2 Research Contributions

BOTHER automatically derives the ordinal relations it uses. These arise during **analytic abduction** either from matching a qualitative behaviour to a function descriptor or from choosing which of a set of mathematical disjuncts to solve first. Other magnitude reasoning systems assume the user specifies these relations explicitly, *via* assertions of the form $X \ll Y$, or implicitly, *via* domain rules [Bennett 87]. **BOTHER**'s ability is a consequence of it integrating qualitative and algebraic reasoning.

A second feature is that **BOTHER** deals with mathematical functions rather than qualitative parameters. Its approximations can therefore be based on the concepts of *functional approximation* and *limit* instead of an algebra of relative magnitudes. In fact by exploiting ordinal relations and truncated Maclaurin expansions in a *quotient variable*, a continuum of functional approximation can be conceived.

Third, **BOTHER** has an explicit *meta-magnitude* level which controls the choice of expansion variable (*i.e.* creation of a *quotient variable*), the *phrasing* (of an expression), and the course of subsequent simplification (*via* the *order* of approximation).

Finally, **BOTHER** can be thought of as extending **PRESS** with an approximate reasoning capability. So whereas in **PRESS** if a subterm cannot be eliminated algebraically it must remain, in **BOTHER**, it is examined to see if it can be approximately bounded. As the original **PRESS** methods depend on such meta-level concepts as the number of occurrences of a variable, the effect of such **BOTHER** inferences is to extend the power of the pure methods.

8.10.3 Conclusions

This chapter has shown how **B**ack-**O**f-**TH**e-**E**nvelope-**R**easoning can greatly reduce the complexity of expressions thereby rendering them more easily comprehended by emphasizing their important functional relationships. We envisage such approximations might be useful in a design or tutoring context where they expose the key features of the domain problem.

Chapter 9

Summary and Conclusions

9.1 Summary of the Techniques

This thesis has developed a number of computational techniques for finding approximate functional solutions to ordinary differential equations. These are mathematical expressions whose qualitative forms and functional relationships embody the most significant features of the real solutions. By using approximate functional solutions we can exploit the best features of both qualitative and mathematical descriptions: namely the comprehensibility of qualitative solutions and the conciseness and predictive utility of mathematical solutions. The interplay of qualitative and mathematical knowledge is the essential element.

Two strategies for approximate functional reasoning were considered. One abstracted the original equation, predicted behaviour in the abstraction space and mapped this back to the approximate functional level. The other solved the equation exactly and then functionally approximated the solution. Each strategy was realised using a set of techniques designed to dovetail together to achieve a style of reasoning closer to that of an engineer or physicist rather than a pure mathematician.

In this chapter, we shall briefly review the essentials of each technique and examine how they address the research issues laid down at the outset. We then point to the research contributions made and suggest directions for further work.

9.1.1 Analytic Abduction

The input to **analytic abduction** consists of an ordinary differential equation whose coefficients are free of the independent variable. This restriction is due to the limitations inherited from the use of Kuipers **QSIM** algorithm [Kuipers 86a].

The goal is to conjecture an approximate functional solution of the original equation.

A rational reconstruction of **QSIM** is used to generate a qualitative behavioural prediction. This is mapped to an equivalence class of parameterized functions whose qualitative behaviours are congruent with the output from **QSIM** (*i.e.* they have the same sequence of critical points, convexities and bounding intervals). The parameterized function is then substituted into the differential equation. If the correct solution had been guessed the equation could be satisfied identically. However, in general, it will give rise to a residual error. Therefore, the conjecture is optimized with respect to the equation by finding a value for the parameter(s) which minimizes the residual error. This stage employs symbolic manipulation methods akin to those in **PRESS** in concert with an aggressive symbolic simplification strategy that reflects the reasoning of real engineers.

Consequently from a differential equation and a purely qualitative simulation it is possible to compose a high level description of behaviour and highlight the approximate functional relationships between the problem parameters. Such an expression might subsequently be used to answer queries about how behavioural characteristics depend on parameters in the original model.

It is impossible to encode a separate trial function for all conceivable qualitative behaviours as there are an infinite number of them. Even restricting consideration to behaviours containing a finite number of critical points is still a daunting task. **Segment calculus** was invented to address this problem.

9.1.2 Segment Calculus

The input to **segment calculus** is a qualitative behavioural prediction of the solution to a differential equation which is not recognised as being a member of any equivalence class in the standard library of known functions.

The goal is to conjecture how the unknown qualitative function could possibly arise as the interaction of two other (simpler) qualitative functions. Subsequently, this information could then be used either in conjunction with **analytic abduction** or **equation parsing**.

To find an interpretation, the input behaviour is partitioned at its critical points into a sequence of monotone segments. Each segment is "explained" in all possible ways as a mathematical combination of pairs of segments consistent with the rules of **segment calculus**. A spanning interpretation is then built by finding concatenatable segment sequences through the sets of possibilities.

In the simplest case, the sequences concatenate to monotone functions. Where they do not concatenation is permitted only if a function is known which has the sequence of segments seen so far (otherwise every segment sequence would be permissible and the system would have little inferential power).

At the very least **segment calculus** can provide a justification for conjecturing a specific *compound* form for the solution of an equation. This conveys partial information about the solution which emphasises its dominant operator as well as its qualitative behaviour. Often, on substituting this into the differential equation, it provides just enough structure to recognise what the solution might be. The following technique is designed to formalise this notion.

9.1.3 Equation Parsing

The input to **equation parsing** is a differential equation and a conjecture as to its compound functional form (*e.g.* the product, sum, composition or exponentiation of two simpler but unknown functions).

The goal is to identify the simpler functions and hence construct a particular solution of the equation.

The compound form is substituted into the differential equation and its derivatives evaluated. For this equation to hold certain combinations of the terms must interact in certain ways *e.g.* a pair of terms may be additive inverses so that their sum is zero or multiplicative inverses so that their product is one. A parse of the equation is created by installing arcs between terms looking for these inverse relations. This is rather like parsing a sentence from a natural language but instead of having purely sequential structures more complex embeddings of arcs are allowed.

It is possible to parse *top down*, looking for all possible parses or *bottom up* from a set of function cliches, which is rather more efficient. Each cliche describes a function in terms of its pattern of recurrent derivatives and function symbols in its first and higher derivatives.

This allows fragments of the equation to be recognised as being characteristic of the behaviour of a certain functions. Under this way of thinking, the goal becomes one of finding a partitioning of the terms in the equation such that the interpretation of the function symbols is *consistent* (each function symbol has a unique interpretation) and *complete* (there are no unaccounted-for terms).

9.1.4 Closed Form Approximation

The previous methods are based on the philosophy of abstracting the original equation, reasoning in the abstraction space and mapping the results back to the mathematical level. However, there is another way to approach the problem of finding approximate functional solutions to differential equations. This consists of solving the equation exactly and then approximating the solution. In fact, this is the approach advocated by Sacks in his **QMR** system [Sacks 85a, 85b]. However, **QMR** can only deal with equations having closed form solutions. We introduced the technique of **closed form approximation** to deal with those equations which do not have closed form solutions.

The input to **closed form approximation** consists of a regular singular differential equation.

The goal is to find a simple mathematical expression which approximates its exact, infinite series, solution in closed form.

The method consists of four stages. First the given equation is solved in terms of an implicit infinite series defined *via* a recurrence structure. Next the recurrence structure is transformed into an equivalent **case** representation. In this form, properties of the series are characterised as sets of numbers defined *via* generators in modular arithmetic. Following this, a set of closed form expressions are assembled out of a library of standard forms and simple polynomials such that each one's infinite series representation is *signature-index* equivalent and *coefficient-sequence* comparable to the exact solution. Finally, the closed form expression which is closest to the exact solution, according to a well defined metric, is identified.

9.1.5 Back-of-the-Envelope Reasoning

Back-of-the-envelope reasoning is a second technique which approximates mathematical expressions directly rather than performing inference over an abstraction of the model from which they are derived. It is used in conjunction with **analytic abduction** to simplify overly complicated formulae in order to render them more comprehensible.

The input to **back-of-the-envelope** reasoning is a mathematical expression and a set of ordinal relations between symbolic constants, of unknown absolute value, mentioned in it.

The task is to calculate rough numerical bounds for the expression even though the absolute values of the constants is unknown.

The method begins by using the ordinal relations to create a composite variable whose magnitude is provably less than one. By rewriting the expression in terms of this "small" variable, it becomes possible to functionally approximate and subsequently bound it.

BOTHER makes the most conservative assumptions necessary about the relative magnitudes of the unknown constants to return a numerical estimate. If subterms in the the expression can be rewritten as Maclaurin series the approximation is chosen to be the lowest order truncation still containing the expansion variable. Otherwise a zero order approximation is made by taking the limit as the expansion variable tends to zero.

The limit evaluator is based on natural deduction rules derived from the analysis of human limit calculations. The current systems knows of five techniques: direct evaluation, completion of the square, factorisation, l'Hôpitals method and the behaviour of the logarithm method.

9.2 Relative Merits

For equations amenable to **closed form approximation** and **equation parsing**, both techniques yield the same result. For those equations where both **analytic abduction** and **closed form approximation** may be used, **closed form approximation** will in general produce more informed and accurate approximations. All three techniques are needed to cover a broad range of problems as no one technique subsumes any other entirely.

Segment calculus and **back-of-the-envelope** reasoning are ancillary techniques. **Segment calculus** extends **analytic abduction** to cases where a trial function of the right qualitative form is not immediately known. **Back-of-the-envelope** reasoning simplifies overly complicated formulae to restore comprehensibility.

The main problem with **equation parsing** is that there is no way of knowing, at the outset, whether or not a closed form solution exists. All **equation parsing**, or any human, can do is to try some heuristic resource-bounded exploration and see if anything emerges. Viewed from a computer science perspective this seems futile. Viewed from an artificial intelligence point of view, **equation parsing** formalizes a model of human reasoning. I suspect this kind of free exploration without guarantees is actually a fairly good model of a great deal of mathematical creativity.

9.3 The Research Issues

In order to perform approximate functional reasoning, a number of research issues have to be addressed as follows:

- *Abstraction*. What elements of a real solution can be abstracted away to make an approximate functional solution?
- *Strategy*. Is it better to create an approximate functional solution *top down* by reasoning directly from the differential equation or *bottom up* by reasoning from the equation's exact solution?
- *Adequacy*. Is a proposed approximation sufficiently accurate (in quantitative terms) for predictive purposes?
- *Fidelity*. How do we ensure that the approximate functional solution is qualitatively equivalent to the real solution?
- *Optimality*. Is there a notion of a *best* approximate functional solution?

9.4 Addressing the Issues

9.4.1 Abstraction

In general the real and approximate functional solutions will share some features but not others. **Analytic abduction** abstracts over the qualitative form of the solution. **Equation parsing** abstracts over the operator decomposition of the solution. **Closed form approximation** abstracts over the coefficient, signature and index sequences of the infinite series representation of the exact solution.

These define a sequence of abstractions of increasing specificity. If **analytic abduction** were *only* to abstract over the qualitative form then its predictions would be poor. Its' power comes from the optimisation of a parameterised function with respect to the differential equation.

The abstraction equation parsing uses is a generalisation of the classic variables separable method for solving partial differential equations.

9.4.2 Strategy

Analytic abduction and **equation parsing** adopt a *top-down* approach. That is, they both begin with the differential equation and construct their solutions by positing constraints on the real solution to derive an analytic function which satisfies them.

Closed form approximation employs a *bottom-up* strategy constructing an approximation from the exact infinite series solution.

We might expect the *bottom up* strategy to lead to better approximations since it has the exact solution explicitly available. However, the technique we currently use for this, **closed form approximation**, makes local, rather than global approximations. In other words its approximations are good in a limited region but not necessarily overall. **Analytic abduction** however, makes global approximations by focusing on the qualitative features of an entire behaviour. So, counter to our expectations, it appears the *top down* approach is perhaps a more useful strategy.

Another *top-down* strategy we have not yet explored is that of functionally approximating the coefficients of the differential equation to create an approximate model. The truncated Maclaurin method, suggested earlier, again only amounts to approximation in a magnitude extreme. An interesting problem for the future is to find approximations which hold over a wider range of values and yet are simpler to work with.

9.4.3 Adequacy

In each of the techniques we have attempted to ensure the adequacy of an approximate functional solution, although the precise mechanisms have varied from case to case.

In **analytic abduction** the approximation methods rest on some fundamental assumptions. For *collocation* it was that the optimum values of the adjustable parameters were insensitive to the exact choice of collocation points. For *harmonic balance* it was that the series whose beginning was represented by the residual equation was rapidly convergent. Hence it was possible to assess the adequacy of the approximate solution by testing these assumptions. This was done either by varying the collocation point and computing the ratio of parameter value predictions, or by computing the ratio of the lowest neglected subterm to that of the restoring term in the original oscillator equation. Both of these techniques are common engineering practice.

The adequacy of the interpretations of **equation parsing** is somewhat easier to assess because of the manner in which a solution is constructed. Any solution found is guaranteed to be a particular exact solution of the original differential equation.

In **closed form approximation** a series is deemed to be an adequate approximation if it is both signature-index equivalent and coefficient-sequence comparable to the infinite series of the exact solution. Otherwise it is an inadequate approximation and is rejected.

9.4.4 Fidelity

It is important that approximate functional solutions have the same gross qualitative form as the exact solution. This property is embodied in the *fidelity* requirement.

As the trial functions of **analytic abduction** are selected by matching function descriptions to a qualitative behavioural prediction they should preserve fidelity. Unfortunately current qualitative reasoners can predict spurious behaviours *i.e.* behaviours which are not realizable in the real system [Kuipers 86a, Struss 88a]. This can confuse functional approximation because it means that the qualitative simulator cannot be relied upon to generate only correct behaviours.

As this problem is being tackled by others in the field we take the pragmatic view of ignoring it in the expectation that a solution will be found (perhaps using a phase space approach or additional filters). If spurious behaviours do arise they are treated as any other:

indeed there is no way of identifying them. However, if a system has a unique behaviour it is guaranteed to be realizable [Kuipers 86a p321]. In practice, spurious behaviours have not proved to be a serious handicap so far. Nevertheless, as we move to more complex systems, they will undoubtedly become troublesome. However, that is a quite separate research issue and in order to circumscribe a realistic programme of work it is fair to isolate it from approximate function reasoning.

Fidelity in **closed form approximation** depends on how close the sequences of coefficients of the infinite series of the exact solution and the infinite series of the approximation remain. To assess this we invented a metrical measure of the "distance" between the first five coefficients of exact and approximate solutions. If this distance is zero, and signature-index equivalence also holds, the approximation is deemed to be faithful to the exact solution. However, in the more general case, the values of the coefficient sequences will diverge and the distance between the coefficient sequences will become non-zero. Consequently, *fidelity* can then only be expected in some limited interval.

Although one could contrive series that share the first few terms and then rapidly diverge we have never come across them in physical problems.

When **equation parsing** succeeds it finds particular exact solutions. Consequently it preserves fidelity.

9.4.5 Optimality

Although it should be possible to define a concept of "best" approximation in **analytic abduction** this is not done. The program terminates as soon as a trial function is found which passes the adequacy criterion. Other, possibly better, approximations are never examined. The rationale behind this strategy is that the symbolic manipulation required to compare and rank two approximations would be very high indeed as each trial function can be very different functionally. Moreover, **analytic abduction** minimises the residual error at only a finite number of (collocation) points. Although it is assumed that the error remains

small in between there is no explicit measure of this. Consequently, the crudity of the method does not warrant the computational expense in finding multiple approximations.

The "best" **closed form approximation** is the one which is both *signature-index* and *coefficient sequence* equivalent to the infinite series of the exact solution. In other words, if the differential equation actually had a closed form solution, involving functions known to the standard library, then closed form approximation will always prefer the exact solution over any other approximation. In general, however, the *coefficient sequences* will only be comparable, not equivalent. In such cases, the "best" approximation would be the *signature-index* equivalent contender whose distance to the exact solution is least.

9.5 Contributions

The original contributions arising out of thesis are:

- *Concept* of *approximate functional solution* as a new level of distinction in describing the behaviour of physical systems.

- *Idea* of exploiting qualitative reasoning to guide mathematical approximation.

- *Model & Algorithm* for realizing these ideas based, in part, on a formalization and computational implementation of Acton & Squire's [Acton & Squire 85] model of engineering inference originally intended for manual application. The new features include, a clarification of the meta-level, automatic behavioural envisionment, behavioural revision, a formalization and automation of trial function invocation and a symbolic manipulation program.

- *Technique & Algorithm* to extend Acton & Squire's original model to deal with cases were the first trial function proves to be inadequate. This utilises the three new concepts of *topological isomorphism, topological similarity* and *exaggeration.*

- *Technique & Algorithm* called *segment calculus* for reasoning about the qualitative effects of segment interaction and its implementation in a program to determine possible interpretations of a qualitative behaviour.

- *Technique & Algorithm* called *back-of-the-envelope* reasoning for heuristic symbolic simplification aimed at revealing the gross underlying structure of an expression.

- *Model & Algorithm* called *equation parsing* of an inspection method for "solving" a differential equation based on a treatment of an equation in an analogous way to a natural language parsing problem. So far as we are aware this is the first link between mathematical reasoning and techniques of computational linguistics. The parsing approach should generalise to other kinds of equations.

- *Technique & Algorithm* for performing *closed form approximation.*

- *Representation* of infinite series which is both novel and computationally efficient based on generators in a modular arithmetic.

- *Concepts* of *signature-index* and *coefficient-sequence* abstractions and their use in defining a *metric* between infinite series.

- *Algorithms* of lesser importance, but necessary for completeness, for performing limit evaluation and series solution of differential equations; the latter appealed for by Sacks. The limit evaluator is implemented in the meta-level style of **PRESS**. The solution in series program is specified but unimplemented. Both algorithms are our own.

- *Rational Reconstruction* of **QSIM** including autonomous mapping from an ordinary differential equation to a minimal set of qualitative constraints and a behavioural revision mechanism to propagate new landmark information back through the envisionment. The latter is necessary when matching to function descriptors.

We are not claiming that approximate functional reasoning is a panacea. Either mathematical or qualitative reasoning can be more apposite depending on the application.

For example, if accuracy is paramount an exact mathematical solution is generally preferable. Whilst if a causal account of behaviour is desired [Kuipers 86b] or the analytic form of the coefficients of the original differential equation are unknown, qualitative reasoning is preferable. In fact the latter problem, which would arise if, for example, certain coefficient functions are known only to be monotonic, is one of the principal justifications for the whole qualitative reasoning enterprise. The point is that qualitative differential equations need not necessarily be derived from ordinary differential equations (although in **analytic abduction** they are).

However, approximate functional reasoning does offer an alternative view of reasoning about physical systems. Moreover, it is a view which is motivated by the cognitive behaviour of real engineers and builds on approximation methods developed over many years.

9.6 Further Research

9.6.1 Analytic Abduction

Analytic abduction could be extended in several ways. First, the initial selection of a trial function is based purely on the predicted qualitative form of the exact solution. This is fairly blind as many distinct classes of function have the same qualitative form. Now, of course, one could argue that this is not relevant as the whole point of **analytic abduction** is to find rough approximations to behaviour. Nevertheless given two trial functions it is possible that one would have been a better approximation than another but that it was never examined because the first trial function happened to pass the adequacy criterion. To overcome this it might be useful to modify the order in which trial functions are tested depending on an analysis of the structure of the original differential equation or by exploiting domain knowledge about the "typicality" of a class of functions with respect to a certain kind of problem.

A second extension would be to explore the idea of critical point suppression introduced in *Chapter 3*. The idea was that in a behaviour containing many critical points some fluctuations are of much greater amplitude and significance than others. Consequently, it might be possible to smooth the behaviour by replacing certain fluctuations with a monotonic region thereby reducing the number of critical points. This would permit matching to analytic forms that would not have been sanctioned if every detail in the qualitative behaviour had been retained. The formalization of this would have to involve the imposition of magnitude relations between the landmarks of a qualitative behaviour.

Finally, the range of problems for which **analytic abduction** is applicable would be increased if qualitative simulation could be extended to differential equations which contain coefficients in the independent variable. In the case of `QSIM` these would be time dependent coefficients corresponding, for example, to forced oscillators. The current difficulty stems from the impoverished temporal representation used so that, in the example just mentioned, the period of the driving force cannot be unambiguously related to the (scale-free) sequence of time points discovered by qualitative simulation.

9.6.2 Equation Parsing

The concept of **equation parsing** should generalise to other mathematical areas which involve commutative operators and for which it is possible to define a notion of inverse relation and abstract "form" of the solution. Commutivity is necessary to allow interaction arcs to interleave in a complex referential pattern. Parsing in equations involving purely non-commutative operators would be closer, in some ways, to conventional natural language parsing as the sequence of permissible interactions would again be more rigid.

Instead of formalizing the parsing in the symbolic paradigm we considered re-implementing the parser on a neural net. The hope was to model cliché fragments as patterns of excitation in the net. The problem with this is that it is not at all clear what to make of near misses in this domain. A near miss in this context would be a function which satisfies most of the constraints with some term is left over: in effect it is only "almost correct". The other approximation methods in this thesis are based on abstractions of the exact solution.

Excitation patterns in the net are not based on abstractions. Consequently I doubt that near miss matches in the net correspond to mathematically useful entities. One could summarise this as "*an almost correct solution is no solution*". An approximate solution differs from an almost correct one by being based on an *abstraction* which retains some useful information.

9.6.3 Closed Form Approximation

Currently, if the indicial indices are equal or differ by an integer the method so far described will only yield one solution. However, the theory exists to say how to obtain a second solution [Kreider *et al* 80] but we have not yet formalized this algorithmically.

A useful extension to **closed form approximation** would be to deduce the interval in which the approximation is within a certain accuracy of the exact solution.

9.6.4 Back-Of-The-Envelope Reasoning

BOTHER could be improved by incorporating dynamic error analysis to assess the accuracy of a given order of approximation. Formulae for bounding the error incurred in truncating a Maclaurin expansion after a finite number of terms are well known [Spiegel 74] but require knowledge of the range of values the expansion parameter may take on. Our pragmatic strategy at the moment has been to use BOTHER only to discover approximations when the expansion parameter is known to have a magnitude between zero and one.

9.6.5 Antagonistic Reasoning

One great advantage of using mathematical formulae as a representation language for a behaviour is that they can be used to answer comparative analysis queries more specifically than qualitative methods alone. For example, given antagonistic perturbations (*e.g.*

changing the mass of a spring and the spring constant simultaneously) such that one effect tends to cancel that of the other, qualitative comparative analysis [Weld 87, Weld 88a, Weld 88b, Weld 88c, Weld 88d] cannot say anything about the effect on the behaviour.

Using approximate functional solutions it is, in principle, possible to return the necessary magnitude distinctions which delimit the range of possible behaviours. An interesting extension to the techniques developed would therefore be a formalization of the reasoning required to draw such inferences.

9.6.6 Partial Differential Equations

Qualitative reasoning with systems specified by partial differential equations still seems some way off. The complication is that the landmarks in orthogonal space dimensions can only be partially ordered. To illustrate consider the following diagrams. These represent the quantity spaces of two orthogonal space dimensions, with landmark values represented by subscripted Ls.

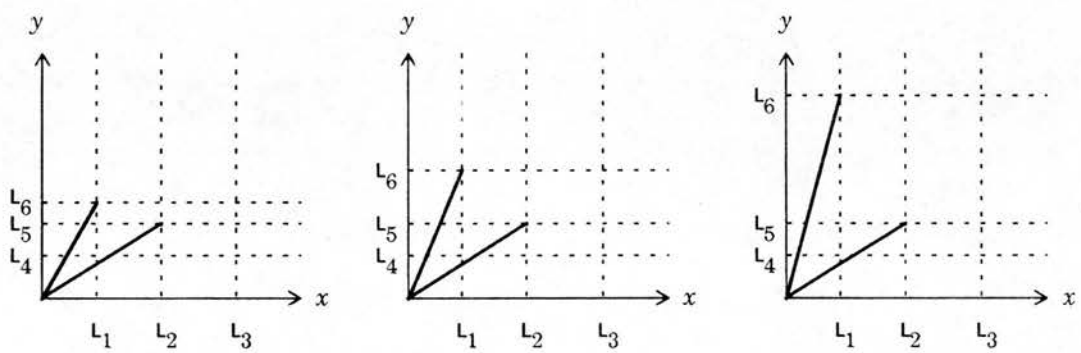


Figure 9-1. Non-metrical Nature of N-Dimensional Quantity Space

The diagrams show three different relative magnitude possibilities for the landmarks in the quantity space of the y dimension. If we define a metric to be the Euclidean distance

$$d(L_x,L_y) = \sqrt{(L_x^2 + L_y^2)}$$

the lack of a quantitative measure of distance between landmarks means that it is not possible to state whether $d(L_2,L_5) > d(L_1,L_6)$, $d(L_2,L_5) = d(L_1,L_6)$ or $d(L_2,L_5) < d(L_1,L_6)$. Any qualitative spatial reasoner would have to consider all possible orderings and this would lead to an intractably large envisionment.

The problem does not arise with time as there is only one time dimension and the time points can be totally ordered.

We mention this because we believe approximate functional reasoning might be a more useful way to proceed. This is because it is sometimes possible to reduce a partial differential equation to a system of ordinary differential equations by, for example, the variables separable method outlined in *Chapter 5*. Each of these could then be subjected to a procedure for approximate solution.

9.6.7 Potential Applications

This dissertation has focused on the theory of approximate functional reasoning rather than on its application. However, we believe approximate functional reasoning offers interesting additional features for both intelligent tutoring systems and design support systems. Consider the following scenarios.

9.6.7.1 Intelligent Tutoring Systems

We envisage a physics tutoring system. Suppose one of the aims of the system is to encourage correct "physical intuitions". This is an elusive quality. In current Intelligent Tutoring Systems [Wenger 87] it is, at best, assumed to be induced by exposure to an apposite problem set. More often, it is ignored in favour of inculcating problem solving schemata.

We want to address a student's intuitions head-on and foresee approximate functional reasoning as a valuable tool for this application. In this scenario, the tutoring system would derive a rough approximate functional solution to some problem and the student would be asked to suggest justifications for both its qualitative features and its functional relationships. This would encourage thought about the relative importance of competing processes as well as general physical principles. Moreover, if the system were then to proceed to solve the problem exactly, the student could then compare his or her intuitive predictions with the exact behaviour.

9.6.7.2 Design Support Systems

Second, in a design support application, we envisage a designer defining a model of a device and submitting it to approximate functional reasoning to determine its principal features. This may prove to be better than numerical techniques as there is an explicit proposal for the principal functional dependencies. Moreover, it may prove to be better than exact symbolic solution as the differential equations corresponding to the device models may be extremely difficult, if not impossible, to solve exactly or perhaps would not possess a closed form solution.

These applications remain speculative at this stage. However, both Acton & Squire [Acton & Squire 85 pp8-10 and p177] and Weld [Weld 87 p959] mention similar ambitions.

9.6.8 Integration of Numerical, Qualitative and Analytic Knowledge

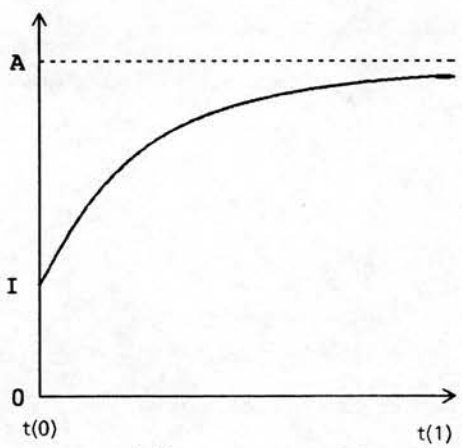
The key message of this thesis is the need for an integration of different levels of reasoning about systems specified using differential equations. Some initial work has been done which combines numerical and qualitative reasoning [Yip 87, Yip 88, Berleant & Kuipers 88] and qualitative and analytic reasoning [Sacks 85a, Sacks 87b, Bennett 87, B.Williams 88] but we believe there is also value in exploiting approximate functional reasoning.

A truly integrated system might begin to realise the original objective laid down by de Kleer in his early **NEWTON** system [de Kleer 75, de Kleer 77] of using simpler methods for solving simpler problems and explaining their reasoning in a human-like way. This thesis presents another facet of this larger picture.

Appendix I

Sample Library Entries

name: rising-exponential



rising exponential

function: `F = A - (A-I)*exp( - b*t)`

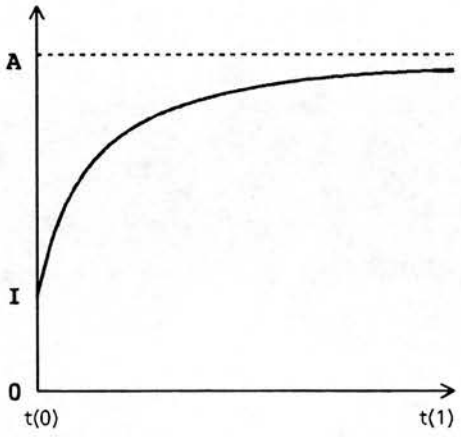
descriptor:
```
{t(0):   [<I,inc>,      <[0,+inf],dec>],
 t(0,1): [<[I,A],inc>, <[0,+inf],dec>],
 t(1):   [<A,std>,      <0,std>]}
```

consequences: {A>I,
asymptotic-approach(F,A),
horizontal-shift(0),
vertical-shift(I),
vertical-scale(A),
is-infinite(t(1))}

parameters: [b]

---

name: rising-hyperbolic-tangent



rising hypebolic tangent

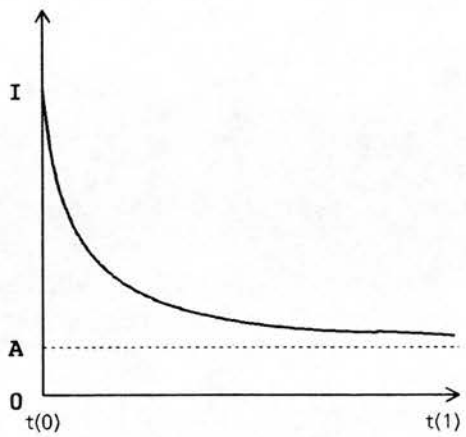
function: `F = I + (A-I)*tanh(b*t)`

descriptor:
```
{t(0):   [<I,inc>,      <[0,+inf],std>],
 t(0,1): [<[I,A],inc>, <[0,+inf],dec>],
 t(1):   [<A,std>,      <0,std>]}
```

consequences: {A>I,
asymptotic-approach(F,A),
horizontal-shift(0),
vertical-shift(I),
vertical-scale(A),
infinite(t(1))}

parameters: [b]

name: falling-exponential



falling exponential

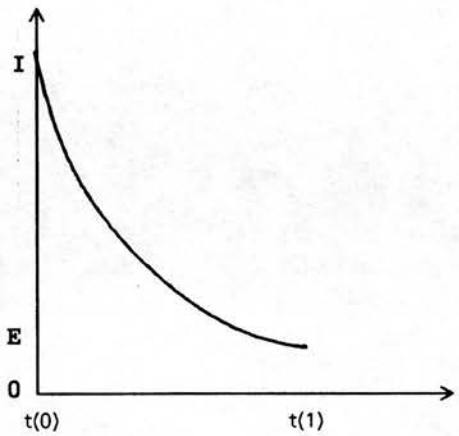
function: `F = A + (I-A)*exp(-b*t)`

descriptor:
```
{t(0):   [<I,dec>,     <[-inf,0],inc>],
 t(0,1): [<[A,I],dec>, <[-inf,0],inc>],
 t(1):   [<A,std>,     <0,std>]}
```

consequences: {**A<I**,
asymptotic-approach(**F**,**A**),
horizontal-shift(**0**),
vertical-shift(**A**),
vertical-scale(**I**),
infinite(**t(1)**)}

parameters: [b]

name: falling-half-parabola



falling half parabola

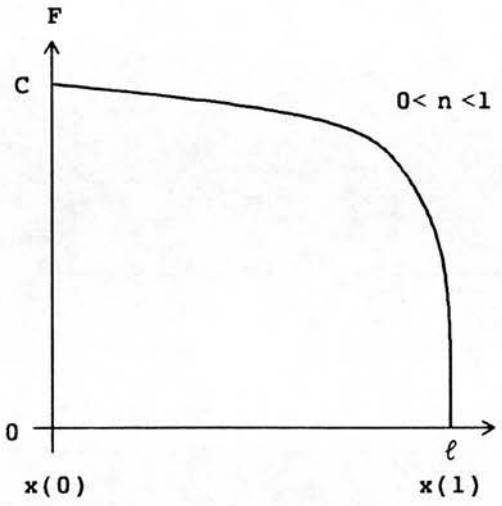
function: $F = I + (E-I)(1-((t-b)/b)^2)$

descriptor: {t(0): [<I,dec>, <[-inf,0],inc>],
t(0,1): [<[E,I],dec>, <[-inf,0],inc>],
t(1): [<E,std>, <0,std>]}

consequences: {E<I,
horizontal-shift(b),
vertical-shift(E),
vertical-scale(I),
finite(t(1))}

parameters: [b]

name: binomial



function: $F = C*(1-(x/\ell))^n$

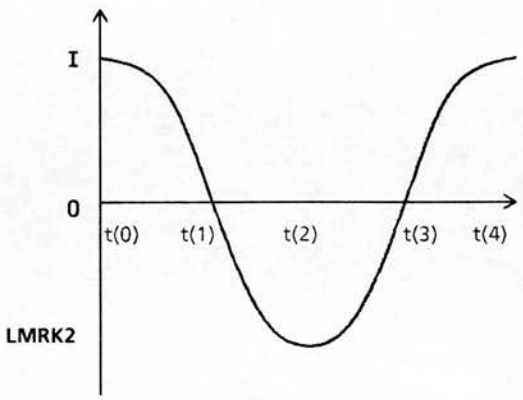
descriptor:

{x(0)	<C, dec>	<[-inf, 0], dec>
x(0,1)	<[0,C], dec>	<[-inf, 0], dec>
x(1)	<0,dec>	<-inf, dec>}

consequences: {0<n<1,
horizontal-shift(0),
vertical-shift(0),
vertical-scale(C),
finite(x(1))}

parameters: [ℓ,n]

name: unshifted-cosine



unshifted cosine

function: `F = I*cos(b*t)`

descriptor:

```
{t(0):   [<I,std>,          <0,dec>],
 t(0,1): [<[0,I],dec>,      <[LMRK1,0],dec>],
 t(1):   [<0,dec>,          <LMRK1,std>],
 t(1,2): [<[LMRK2,0],dec>, <[LMRK1,0],inc>],
 t(2):   [<LMRK2,std>,      <0,inc>],
 t(2,3): [<[LMRK2,0],inc>, <[0,LMRK3],inc>],
 t(3):   [<0,inc>,          <LMRK3,std>],
 t(3,4): [<[0,I],inc>,      <[0,LMRK3],dec>],
 t(4):   [<I,std>,          <0,dec>]}
```

consequences: {`I = – LMRK2`,

horizontal-shift(`0`),

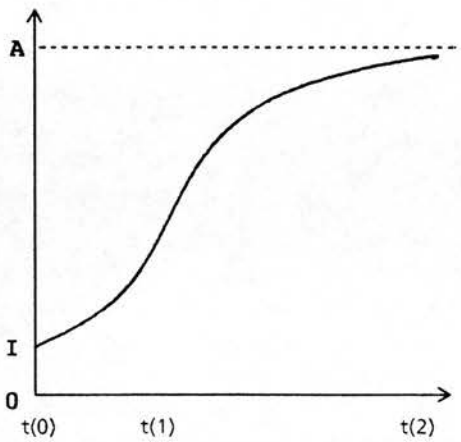
vertical-shift(`0`),

vertical-scale(`I`),

period(`t(4)`)}

parameters: [b]

name: logistic-growth-curve



logistic growth curve

function: `F = A/(1+exp(-a*(t-b))`

descriptor:

```
{t(0):   [<I,inc>,          <[0,LMRK1],inc>],
 t(0,1): [<[I,A],inc>,      <[0,LMRK1],inc>],
 t(1):   [<[I,A],inc>,      <LMRK1,std>],
 t(1,2): [<[I,A],inc>,      <[0,LMRK1],dec>],
 t(2):   [<A,std>,          <0,std>]}
```

consequences: {`I = 1/(1+exp(a*b))`

asymptotic-approach(**F**,**A**),

horizontal-shift(**0**),

vertical-shift(**0**),

vertical-scale(**A**)}

parameters: [a,b]

Appendix II

Rewrite Rules for Qmodular Arithmetic

II.1 Lemmas

In the following proofs *int*, *intp*, *rat* and *ratp* correspond to types "*integer*", "*positive integer*", "*rational*" and "*positive rational*" respectively.

Lemma 1

$\forall x{:}rat\ \forall z{:}ratp\ \forall y{:}intp\ \forall n{:}ratp$, if $ny \in intp$

$$x \text{ Qmod } y = z \leftrightarrow nx \text{ Qmod } ny = nz$$

Proof ($\rightarrow$)

$\exists N{:}int$

$$x = Ny + z \qquad \text{(def'n of Qmod)}$$

$$\therefore \quad nx = N(ny) + nz$$

but ny is a positive integer so we can write $nx = N(ny) + nz$ as

$$nx \text{ Qmod } ny = nz$$

Proof ($\leftarrow$)

$\exists M{:}int$

$$nx = M(ny) + nz \qquad \text{(def'n of Qmod)}$$

$$\therefore \quad x = My + z$$

but y is a positive integer so we can write

$$x \text{ Qmod } y = z$$

□

Lemma 2

$\forall x{:}rat\ \forall z{:}ratp\ \forall y{:}intp\ \forall n{:}ratp$

$$x \text{ Qmod } y = z \quad \leftrightarrow \quad (x+n) \text{ Qmod } y = (z+n) \text{ Qmod } y$$

Proof ($\rightarrow$)

L.H.S. $\exists N{:}int$

$$x = Ny + z \qquad \text{(def'n of Qmod)}$$

$$\therefore \quad x+n = Ny + z + n$$

Consider the disjoint possibilities of $z+n$ being greater or less than y. Suppose $0 \leq z+n < y$ then $z+n = z+n$ Qmod y so

$$(x+n) \text{ Qmod } y = (z+n) \text{ Qmod } y$$

□

Suppose $y \leq z+n$ then $\exists P{:}int\ \exists r{:}rat\ (0 \leq r < y)$ such that

$$z+n = Py + r$$

$$\therefore \quad z = Py + r - n$$

Hence $x+n = Ny + z + n$ can be written as

$$x+n = Ny + Py + r - n + n$$

$$\equiv \quad x+n = (N+P)y + r$$

And so

$$(x+n) \text{ Qmod } y = r = (z+n) \text{ Qmod } y$$

□

Proof ($\leftarrow$)

R.H.S. $\exists M{:}int$

$$x+n = My + (z+n) \text{ Qmod } y \qquad \text{(def'n of Qmod)}$$

rearranging

$$x = My + (z+n) \text{ Qmod } y - n$$

let $(z+n)$ Qmod $y = r$ then $\exists P{:}int$ such that

$$z+n = Py + r$$

$$\therefore \quad r = z+n-Py$$

$$\therefore \quad x = (M-P)y + z + n - n = Ty + z$$

which implies

$$x \text{ Qmod } y = z$$

□

Lemma 3

$\forall x{:}rat\ \forall z{:}ratp\ \forall y{:}intp$

$$x \text{ Qmod } y = z \ \rightarrow \ (-x) \text{ Qmod } y = (y-z) \text{ Qmod } y$$

Proof

$\exists N{:}int$

$$x = Ny + z \qquad \text{(def'n of Qmod)}$$

$$\therefore \quad -x = -Ny + (-z)$$

$$\therefore \quad -x + (y-z) \text{ Qmod } y + z = -Ny + (y-z) \text{ Qmod } y$$

But we know from the definition of Qmod that $0 \leq z < y$. Consider the possibilities. Suppose $z = 0$ then

$$-x + (y-z) \text{ Qmod } y + z = -Ny + (y-z) \text{ Qmod } y$$

becomes

$$-x + 0 + 0 = -Ny + 0$$

$$\therefore \quad (-x) \text{ Qmod } y = (y-z) \text{ Qmod } y$$

□

Conversely, suppose $0 < z < y$, then

$$(y-z) \text{ Qmod } y + z = y - z + z = y$$

$$\therefore \quad -x = (-N-1)y + (y-z) \text{ Qmod } y$$

$$\therefore \quad (-x) \text{ Qmod } y = (y-z) \text{ Qmod } y$$

□

Lemma 4

$\forall x{:}rat\ \forall y{:}intp\ \forall n{:}int$

$$(x + ny) \text{ Qmod } y = x \text{ Qmod } y$$

Proof

$\exists N{:}int$, $\exists z{:}rat\ 0 \leq z < y$ such that

$$x + ny = Ny + z \qquad \text{(def'n of Qmod)}$$

$$\therefore \quad x = (N-n)y + z$$

But as $0 \leq z < y$ this is equivalent to

$$x \text{ Qmod } y = z$$

□

II.2 Rewrite Rules for Generators

Generators may be manipulated according to the following rules:

II.2.1 Match Rule

Let i_1 and j_1 be the lowest two values spawned by their respective generators. Then the match rule is

$$\{i: i \text{ Qmod } p = q \wedge i \geq \alpha\} \equiv \{j: j \text{ Qmod } p = q \wedge j \geq \beta\} \text{ iff } i_1 = j_1.$$

Two such generators are guaranteed to match if $\alpha - \beta = 0$ and guaranteed not to match if $|\alpha - \beta| \geq p$

II.2.2 Shift Rule

$$\{i+\alpha: i \text{ Qmod } p = q \wedge i \geq \beta\} \equiv \{j: j \text{ Qmod } p = (\alpha+q) \text{ Qmod } p \wedge j \geq (\alpha+\beta)\}$$

Proof

$$\{i+\alpha: i \text{ Qmod } p = q \wedge i \geq \beta\}$$

rename the variable $j = i + \alpha$, then

$$\{j: j-\alpha \text{ Qmod } p = q \wedge j-\alpha \geq \beta\}$$

Using lemma 2 this becomes

$$\{j: j \text{ Qmod } p = (\alpha+q) \text{ Qmod } p \wedge j \geq (\alpha+\beta)\}$$

□

II.2.3 Conjunction Rule

Given two generators in qmodular arithmetic we are required to find the generator such that the sequence of numbers it spawns is precisely the intersection of those of the original pair of generators. This is then the sequence of numbers spawned by the simultaneous conjunction of the two generators. Formally,

$$\{i: i \,\text{Qmod}\, p = q \wedge i \geq \alpha\} \wedge \{i: i \,\text{Qmod}\, p' = q' \wedge i \geq \beta\}$$

$$\equiv \{i: i \,\text{Qmod}\, \text{lcm}(p, p') = \text{rmdr}(q, q', p, p') \wedge i \geq \max(\alpha, \beta)\}$$

where "lcm" returns the least common multiple of its arguments and "rmdr" is defined recursively by

$$\text{rmdr}(q, q', p, p') = \begin{cases} \text{undefined} & \text{iff } q^*q' > \text{lcm}(p, p') \\ 0 & \text{iff } q = 0 \wedge q' = 0 \\ \text{rmdr}((q-1)\text{Qmod}\, p, (q'-1)\text{Qmod}\, p', p, p') + 1 & \text{otherwise} \end{cases}$$

This formula was obtained by induction from a number of examples of what we term "congrugence conjunction tables". Such a table is shown in Table II-1.

x Qmod 9 = \ x Qmod 7 =

	0	1	2	3	4	5	6
0	0	36	9	45	18	54	27
1	28	1	37	10	46	19	55
2	56	29	2	38	11	47	20
3	21	57	30	3	39	12	48
4	49	22	58	31	4	40	13
5	14	50	23	59	32	5	41
6	42	15	51	24	60	33	6
7	7	43	16	52	25	61	34
8	35	8	44	17	53	26	62

Table II-1 Congruence Conjunction Table

The rows of a congruence conjunction table are labelled with all possible integral values of q ($0 \leq q < p$) and the columns are labelled with all possible integral values of q' ($0 \leq q' < p'$). The entries are the values of the remainder such that both the row and column congruences will simultaneously hold.

Having constructed many such tables for arbitrarily chosen pairs of congruences it appears that there is a pattern to the distribution of values for the table entries. The value of the entry at the (q,q')-th square is one less than the number of steps it takes to walk from the top left corner of the table to the square along a certain metric which is defined as follows:

- start at the square (0,0)
- walk down the 45 degree diagonal until either the right edge is hit (in which case increment q and move to the left edge) or the bottom edge is hit (in which case increment q' and move to the top edge)
- proceed again down the 45 degree diagonal repeating the operations whenever an edge is hit.

The formula simply counts the steps along this metric.

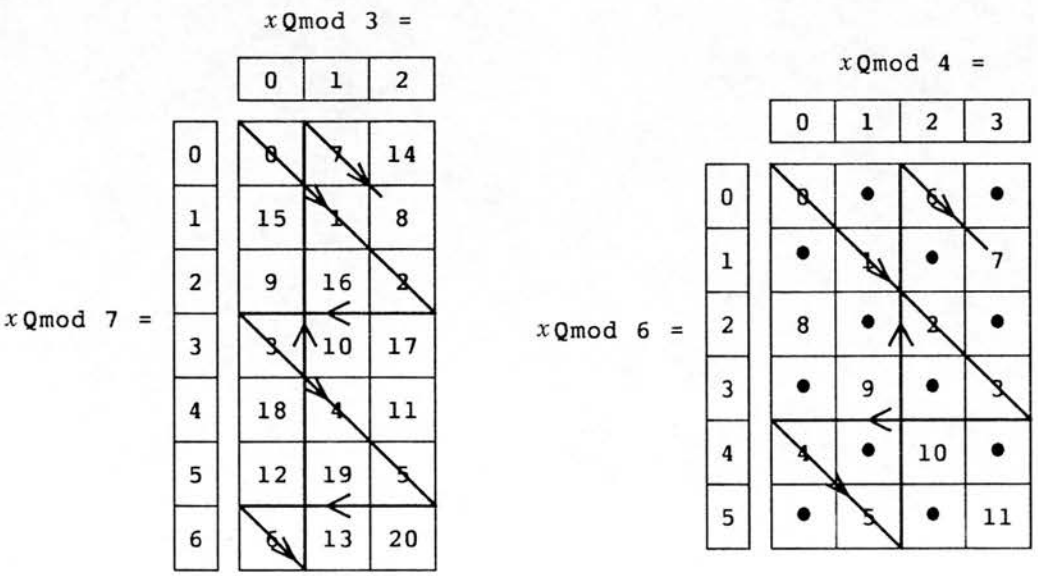


Table II-2 Coprime and Non-coprime Congruence Conjunctions

In the simple cases where the least common multiple of p and p' is their product $p*p'$ every square is accessible. However, if p and p' have a common factor certain squares are inaccessible *via* the metric. This corresponds to mathematically impossible conjunctions of congruences. An example of each type is shown in Table II-2.

A walk through a congruence conjunction table provides a geometric picture of the origin of the formula for the remainder. However, we would have preferred to have derived this algebraically from number theory. This concern prompted a search for an alternative formulation.

Below we outline an alternative approach suggested by Peter Ross [Ross 89: personal communication] which explains how to conjoin congruences using the *Chinese Remainder Theorem* (see [Baker 84 pp18-19]). The difference between the two methods is that Ross's approach yields a solution in terms of a set of constraints which must then be solved by generate & test to calculate the remainder whereas ours computes the required remainder directly.

In the following section we adopt the standard notation for congruences *i.e.*

$$x \equiv y \,(\mathrm{mod}\, z) \quad \Leftrightarrow \quad x \,\mathrm{qmod}\, z = y$$

for x and y integers and z a positive integer.

II.2.3.1 Chinese Remainder Theorem

If m_1, m_2, ..., m_n are relatively coprime in pairs there exists a unique solution modulo $m_1*m_2*...*m_n$ to the n simultaneous congruences

$$x \equiv a_i \,(\mathrm{mod}\, m_i), \quad i = 1,2, ..., n.$$

If any pair is not coprime then they must have a highest common factor > 1. In this case it is necessary to split the congruences into a larger set.

For example, given

$$x \equiv 1 \,(\text{mod } 6) \;\wedge\; x \equiv 4 \,(\text{mod } 15)$$

hcf(6, 15) = 3. Now $x \equiv 1 \,(\text{mod } 6) \Rightarrow x \equiv 1 \,(\text{mod } 2) \;\wedge\; x \equiv 1 \,(\text{mod } 3)$. Similarly, $x \equiv 4 \,(\text{mod } 15) \Rightarrow x \equiv 1 \,(\text{mod } 3) \wedge x \equiv 4 \,(\text{mod } 5)$. Hence, the original conjunction implies

$$x \equiv 1 \,(\text{mod } 2) \;\wedge\; x \equiv 1 \,(\text{mod } 3) \wedge x \equiv 4 \,(\text{mod } 5).$$

But now the *Chinese Remainder Theorem* applies directly (as 2, 3 and 5 are coprime in pairs) and

$$x \equiv 19 \,(\text{mod } 30).$$

I suspect the *Chinese Remainder Theorem* and the recursive formula we derived for the "rmdr" function are, in some sense, equivalent: the latter being a more pictorial representation of the former. However, we have so far been frustrated in all attempts to derive the "rmdr" function by any means other than induction from examples. For the purpose of **closed form approximation**, however, the important point is that it is possible either to compute the solution of a set of simultaneous congruences or determine that they are unsatisfiable. Whether the "rmdr" function or *Chinese Remainder Theorem* is used for this purpose is a matter of personal aesthetics.

II.2.3.2 The Case of Rational Remainders

The operations described above are readily extended to the full qmodular arithmetic by using Lemma 1 to rewrite any qmodular congruences to equivalent modular ones, determining the generator for the conjunction and rewriting this in qmodular format. For example, suppose the task was to conjoin

$$\{i{:}\ i \,\text{Qmod}\, 1 = {}^1/_2 \wedge i \geq 0\} \wedge \{i{:}\ i \,\text{Qmod}\, 2 = {}^3/_2 \wedge i \geq 0\}.$$

These would be rewritten as

$$\{j{:}\ j \,\text{Qmod}\, 2 = 1 \wedge j \geq 0\} \wedge \{j{:}\ j \,\text{Qmod}\, 4 = 3 \wedge j \geq 0\}$$

conjoined to form

$$\{j: j \text{ Qmod } \text{lcm}(2, 4) = \text{rmdr}(1, 3, 2, 4) \wedge j \geq \max(0, 0)\}$$

and simplified to

$$\{j: j \text{ Qmod } 4 = 3 \wedge j \geq 0\}.$$

Finally, as $j = 2i$ this would be rewritten to the equivalent qmodular generator

$$\{i: i \text{ Qmod } 2 = 3/2 \wedge i \geq 0\}.$$

Appendix III

Mapping Recurrences to Case Descriptions

In § 7.3 we asserted that we could rewrite the formula for determining the sign of the ith coefficient as a condition on i. Recall the notation: σ is the sign of the recurrence relation which determines the pattern of the signs of the coefficients in the infinite series; σ_i is the sign of coefficient b_i and $\{k^s_j\}$ are the set of selectors for indicial index s. For simplicity suppose there is only one selector, k. The generalisation to the case with multiple selectors is merely a repitition of the proof for the single case for each one. The formula for σ_i was

$$\{\sigma_i = \sigma^{i-k \,\mathrm{Qdiv}\, n} : i \,\mathrm{Qmod}\, n = k\}$$

where $x \,\mathrm{Qdiv}\, y = [x - (x \,\mathrm{Qmod}\, y)]/y$. Consider the proof in stages:

Case 1: $\sigma = +1,\ \sigma_i = +1$

First suppose $\sigma = +1$ then σ_i will be positive regardless of the index to which σ is raised. Hence,

$$\{\sigma_i = +1 : i \,\mathrm{Qmod}\, n = k\}$$

□

Case 2: $\sigma = -1,\ \sigma_i = +1$

Next suppose $\sigma = -1$. σ_i will be $+1$ whenever $(i\text{-}k)$ Qdiv n is even *i.e.* whenever

$$((i\text{-}k) \text{ Qdiv } n) \text{ Qmod } 2 = 0.$$

But we also know that for σ_i to exist i Qmod $n = k$. Therefore,

$$(i - k) \text{ Qmod } n = 0 \qquad \text{(lemma)}$$

which simply says that $i - k$ is an exact integer multiple of n. Let this multiple be N. Then

$$(i - k) \text{ Qdiv } n = N \text{ and } i - k = N*n.$$

For $(i - k)$ Qdiv n to be even, $N \bmod 2 = 0$ which implies

$$N*n \text{ Qmod } 2n = 0 \qquad \text{(lemma)}$$

or equivalently,

$$(i - k) \text{ Qmod } 2n = 0.$$

Using a lemma again, this implies

$$\{\sigma_i = +1\text{: } i \text{ Qmod } 2n = k\}$$

which completes the proof for the case $\sigma = -1$. $\sigma_i = +1$.

□

Case 3: $\sigma = -1, \sigma_i = -1$

Finally, suppose $\sigma = -1$. In this case for σ_i to be negative, $(i\text{-}k)$ Qdiv n must be odd so

$$((i - k) \text{ Qdiv } n) \text{ Qmod } 2 = 1$$

Again $i\text{-}k$ is an exact integer multiple, N, of n and hence

$$N \text{ Qmod } 2 = 1$$

whereupon $N*n$ Qmod $2*n = n$ and hence

$$\{\sigma_i = +1\text{: } i \text{ Qmod } 2n = n+k\}$$

which completes the proof.

□

Appendix IV

Proofs of Sign Case Mappings

In § 7.3 we claimed rules for the transformation of the *sign case* of a **base** series under composition and multiplication with cx^m. We now prove these results.

IV.1 Proofs of Sign Case Mappings under Composition

We consider the mapping

$$\sum_{i=0}^{\infty} a_i x^i \circ cx^m = \sum_{i=0}^{\infty} a_i c^i x^{im} = \sum_{j=0}^{\infty} b_j x^{j+s}$$

Let the *sign case* of the **base** be

sign: $\{+1: i \text{ Qmod } p = q \wedge i \geq 0\}$ (if the base series only contains positive coefficients)

or

sign: $\{+1: i \text{ Qmod } 2p = q \wedge i \geq 0\} \wedge \{-1: i \text{ Qmod } 2p = p+q \wedge i \geq 0\}$ (coeffs. alternate)

Our goal is to map the *sign case* of the **base** to that of the **candidate** and then to write the **candidate** in terms of **target** parameters. In an application we would then test whether the *sign cases* of **candidate** and **target** match. By writing the **candidate** *sign case* in terms of **target** parameters we can make the matching process easy as some possibilities are seen to be universally true *e.g.* composing or multiplying with a polynomial in which c is positive

leads to a **candidate** *sign case* which is syntactically identical to that of the **target** even without evaluating its arguments.

We are able to link the parameters of **base** and **target** *via* Theorem 7.1 which says that to guarantee at least index equivalence $mp = n$ and $mq = s+k$. The manner in which the *sign case* transforms under composition depends on both the sign of c in cx^m and on whether a particular i is even or odd. If i is to be even we can represent this as the constraint i Qmod $2 = 0$ conjoined with the generator and then use the rewrite rules for generators developed in *Appendix II* to merge the two propositions together. Similarly for i odd. The various possibilities are enumerated below.

Rule 1: coefficients all positive, $c > 0$

This is the simplest case to consider. We start with a series all of whose terms are positive and compose it with a polynomial whose leading coefficient is positive. Hence the signs of the terms are unaffected.

signs: $\{+1: i \text{ Qmod } p = q\} \longmapsto$

$\{+1*c^i: i \text{ Qmod } p = q\}$ (def'n of composition)

but as $c>0 \rightarrow c^i > 0 \; \forall \, i$

$\therefore$ $\{+1: i \text{ Qmod } p = q\}$

$\{+1: i \text{ Qmod } n/m = (s+k)/m\}$ (theorem 7.1)

$\{+1: im \text{ Qmod } n = s+k\}$ (lemma)

$\{+1: (im - s) \text{ Qmod } n = k \text{ Qmod } n\}$ (lemma)

Moreover, $im = j+s$ (def'n of composition)

$\{+1: j \text{ Qmod } n = k\}$

Hence the *sign case* of the **candidate** written in terms of the **target** parameters is

signs: $\{+1: j \text{ Qmod } n = k\}$

□

Rule ∘2: coefficients all positive, $c < 0$

The complication here is that as c is negative the sign of the coefficient of the **candidate** series will depend on whether the index of the **base** is even or odd.

signs: $\{+1: i \text{ Qmod } p = q\} \longmapsto$

$\{+1*c^i: i \text{ Qmod } p = q\}$ (def'n of composition)

$\{+1: i \text{ Qmod } p = q \wedge i \text{ Qmod } 2 = 0\} \vee$

$\{-1: i \text{ Qmod } p = q \wedge i \text{ Qmod } 2 = 1\}$

using the results of **rule ∘1**

$\{+1: j \text{ Qmod } n = k \wedge (j+s)/m \text{ Qmod } 2 = 0\} \vee$

$\{-1: j \text{ Qmod } n = k \wedge (j+s)/m \text{ Qmod } 2 = 1\}$

$\{+1: j \text{ Qmod } n = k \wedge j \text{ Qmod } 2m = (-s) \text{ Qmod } 2m\} \vee$

$\{-1: j \text{ Qmod } n = k \wedge j \text{ Qmod } 2m = (m-s) \text{ Qmod } 2m\}$

merging the two propositions, the *sign case* of the **candidate** written in terms of the **target** parameters is

signs: $\{+1: j \text{ Qmod } \text{lcm}(n,2m) = \text{rmdr}(k, (-s) \text{ Qmod } 2m, n, 2m)\} \vee$

$\{-1: j \text{ Qmod } \text{lcm}(n,2m) = \text{rmdr}(k, (m-s) \text{ Qmod } 2m, n, 2m)\}$

where lcm(n,$2m$) is the least common multiple of n and $2m$ and the "rmdr" function is as defined in *Appendix II*.

□

Rule ∘3: coefficients alternate, $c > 0$

If the signs of the **base** series coefficients alternate their *sign case* must consist of a disjunction of generators. However, as c is positive, the coefficients of the candidate assume the signs of the corresponding **base** coefficients.

signs: $\{+1: i \text{ Qmod } 2p = q\} \vee \{-1: i \text{ Qmod } 2p = p+q\} \longmapsto \{+1*c^i: i \text{ Qmod } 2p = q\} \{-1*c^i: i \text{ Qmod } 2p = p+q\}$

$\{+1: i \text{ Qmod } 2p = q\}$

$\{-1: i \text{ Qmod } 2p = p+q\}$

$\{+1: j \text{ Qmod } 2mp = (mq - s) \text{ Qmod } 2mp\}$

$\{-1: j \text{ Qmod } 2mp = (mp + mq - s) \text{ Qmod } 2mp\}$

$\{+1: j \text{ Qmod } 2n = k \text{ Qmod } 2n\}$

$\{-1: j \text{ Qmod } 2n = (n + k) \text{ Qmod } 2n\}$

As $0 \leq k < n$, the sign case of the **candidate** becomes

signs: $\{+1: j \text{ Qmod } 2n = k\}$

$\{-1: j \text{ Qmod } 2n = n + k\}$

□

Rule ◦**4**: coefficients alternate, $c < 0$

signs: $\{+1: i \text{ Qmod } 2p = q\} \vee \quad \{-1: i \text{ Qmod } 2p = p+q\} \quad \longmapsto \quad \{+1*c^i: i \text{ Qmod } 2p = q\} \vee \quad \{-1*c^i: i \text{ Qmod } 2p = p+q\}$

which becomes

$\{+1: i \text{ Qmod } 2p = q \wedge i \text{ Qmod } 2 = 0\} \vee$

$\{+1: i \text{ Qmod } 2p = p+q \wedge i \text{ Qmod } 2 = 1\} \vee$

$\{-1: i \text{ Qmod } 2p = q \wedge i \text{ Qmod } 2 = 1\} \vee$

$\{-1: i \text{ Qmod } 2p = p+q \wedge i \text{ Qmod } 2 = 0\}$

$2p$ is guaranteed to be even and p and q are whole numbers. Even and odd numbers obey the following qmodular arithmetic rules:

$i \text{ Qmod } even = even \quad \rightarrow \quad \forall i \text{ even}(i)$

$i \text{ Qmod } even = odd \quad \rightarrow \quad \forall i \text{ odd}(i)$

Therefore, the $\{i\}$ induced by this generator will either be all even or all odd. Hence only two of the four possibilities above will be satisfiable. It is worthwhile, however, to phrase the

mapping in this general form because it will allow us to extend the base series we can have in our standard library to those whose sign pattern alternation is more complex than simply $\{+,-,+,-,...\}$. This may prove useful in the future.

Using previous results, the possibilities above collapse to

$$\{+1: j \text{ Qmod } 2n = k \wedge j \text{ Qmod } 2m = (-s) \text{ Qmod } 2m\}$$

$$\{+1: j \text{ Qmod } 2n = n+k \wedge j \text{ Qmod } 2m = (m-s) \text{ Qmod } 2m\}$$

$$\{-1: j \text{ Qmod } 2n = k \wedge j \text{ Qmod } 2m = (m-s) \text{ Qmod } 2m\}$$

$$\{-1: j \text{ Qmod } 2n = n+k \wedge j \text{ Qmod } 2m = (-s) \text{ Qmod } 2m\}$$

which can be merged to yield

signs: $\{+1: j \text{ Qmod } \text{lcm}(2n,2m) = \text{rmdr}(k, (-s) \text{ Qmod } 2m, 2n, 2m)\}$ (1)

$\{+1: j \text{ Qmod } \text{lcm}(2n,2m) = \text{rmdr}(n+k, (m-s) \text{ Qmod } 2m, 2n, 2m)\}$ (2)

$\{-1: j \text{ Qmod } \text{lcm}(2n,2m) = \text{rmdr}(k, (m-s) \text{ Qmod } 2m, 2n, 2m)\}$ (3)

$\{-1: j \text{ Qmod } \text{lcm}(2n,2m) = \text{rmdr}(n+k, (-s) \text{ Qmod } 2m, 2n, 2m)\}$ (4)

where only two of these will be satisfiable. It is easy to tell which pair it will be by determining whether $p+q$, *etc* are *even* or *odd* from

$$\begin{aligned} even &+ even = even \\ even &+ odd = odd \\ odd &+ odd = even \end{aligned}$$

We can then construct a simple table whose entries are the pair of satisfiable generators given the parity of p and q.

q \ p	*even*	*odd*
even	(1) $\wedge$ (4)	(1) $\wedge$ (2)
odd	(2) $\wedge$ (3)	(3) $\wedge$ (4)

□

IV.2 Proofs of Sign Case Mappings under Multiplication

In this section we provide similar proofs for the transformation of the *sign case* of a **base** series under multiplication with a polynomial cx^m. In this case the relevant mapping is

$$\sum_{i=0}^{\infty} a_i x^i * cx^m = \sum_{i=0}^{\infty} a_i c x^{i+m} = \sum_{j=0}^{\infty} b_j x^{j+s}$$

Let the *sign case* of the **base** be

sign: $\{+1: i \text{ Qmod } p = q \wedge i \geq 0\}$ (if the base series only contains positive coefficients)

or

sign: $\{+1: i \text{ Qmod } 2p = q \wedge i \geq 0\} \wedge \{-1: i \text{ Qmod } 2p = p+q \wedge i \geq 0\}$ (coeffs. alternate)

Our goal is to map the *sign case* of the **base** to that of the **candidate** and then to write the **candidate** in terms of **target** parameters (in an application we would then test whether the *sign cases* of **candidate** and **target** match, but this does not concern us here).

We are able to link the parameters of base and target *via* Theorem 7.2 which says that to guarantee at least index equivalence $p = n$ and $m = s + k - q$. The manner in which the *sign case* transforms under multiplication depends only on the sign of c in cx^m The various possibilities are enumerated below.

Rule *1: coefficients all positive, $c > 0$

signs: $\{+1: i \text{ Qmod } p = q\} \longmapsto \{+1*c: i \text{ Qmod } p = q\}$

As c is positive, this is simply

$$\{+1: i \text{ Qmod } p = q\}$$

But $p=n$, $m=s+k-q$, and $i+m=j+s$

$$\{+1: j+q-k \text{ Qmod } n = q\}$$

$$\{+1: j \text{ Qmod } n = k \text{ Qmod } n\}$$

but as $k < n$

signs: $\{+1: j \text{ Qmod } n = k\}$

□

Rule *2: coefficients all positive, $c < 0$

The proof is as above

signs: $\{+1: i \text{ Qmod } p = q\} \longmapsto \{-1: j \text{ Qmod } n = k\}$

□

Rule *3: coefficients alternate, $c > 0$

signs: $\{+1: i \text{ Qmod } 2p = q\} \quad \{+1*c: i \text{ Qmod } 2p = q\}$

$\{-1: i \text{ Qmod } 2p = p+q\} \quad \{-1*c: i \text{ Qmod } 2p = p+q\}$

$\longmapsto$

$\{+1: i \text{ Qmod } 2p = q\}$

$\{-1: i \text{ Qmod } 2p = p+q\}$

$\{+1: j+s-m \text{ Qmod } 2n = q\}$

$\{-1: j+s-m \text{ Qmod } 2n = n+q\}$

$\{+1: j+q-k \text{ Qmod } 2n = q\}$

$\{-1: j+q-k \text{ Qmod } 2n = n+q\}$

$\{+1: j \text{ Qmod } 2n = k \text{ Qmod } 2n\}$

$\{-1: j \text{ Qmod } 2n = n+k \text{ Qmod } 2n\}$

signs: $\{+1: j \text{ Qmod } 2n = k\}$

$\{-1: j \text{ Qmod } 2n = n+k\}$

□

Rule *4: coefficients alternate, $c < 0$

As above, merely inverting the sign.

signs: $\{+1: i \text{ Qmod } 2p = q\} \quad \{-1: j \bmod 2n = k\}$

$\{-1: i \text{ Qmod } 2p = p+q\} \quad \{+1: j \bmod 2n = n+k\}$

$\longmapsto$

□

Appendix V

Library of Base Series

Series: $\cos x = 1 - x^2/2! + x^4/4! - x^6/6! + \ldots$

Index: $\{i: i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

Coeff: $\{a_i: i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

Sign: $\{+1: i \text{ Qmod } 4 = 0 \wedge i \geq 0\} \vee \{-1: i \text{ Qmod } 4 = 2 \wedge i \geq 0\}$

Mgtde: $\{1, 1/2, 1/24, 1/720, 1/40320\}$

Series: $\sin x = x - x^3/3! + x^5/5! - x^7/7! + \ldots$

Index: $\{i: i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Coeff: $\{a_i: i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Sign: $\{+1: i \text{ Qmod } 4 = 1 \wedge i \geq 0\} \vee \{-1: i \text{ Qmod } 4 = 3 \wedge i \geq 0\}$

Mgtde: $\{1, 1/6, 1/120, 1/5040, 1/362880\}$

Series: $\exp x = 1 + x + x^2/2! + x^3/3! + \ldots$

Index: $\{i: i \text{ Qmod } 1 = 0 \wedge i \geq 0\}$

Coeff: $\{a_i: i \text{ Qmod } 1 = 0 \wedge i \geq 0\}$

Sign: $\{+1: i \text{ Qmod } 1 = 0 \wedge i \geq 0\}$

Mgtde: $\{1, 1, 1/2, 1/6, 1/24\}$

Series: $\exp x^2 = 1 + x^2 + x^4/2! + x^6/3! + \ldots$

Index: $\{i: i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

Coeff: $\{a_i: i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

Sign: $\{+1: i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

Mgtde: $\{1, 1, 1/2, 1/6, 1/24\}$

Series: $\ln(1 + x) = x - x^2/2 + x^3/3 - \ldots$

Index: $\{i: i \text{ Qmod } 1 = 0 \wedge i \geq 1\}$

Coeff: $\{a_i: i \text{ Qmod } 1 = 0 \wedge i \geq 1\}$

Sign: $\{+1: i \text{ Qmod } 2 = 1 \wedge i \geq 1\} \vee \{-1: i \text{ Qmod } 2 = 0 \wedge i \geq 1\}$

Mgtde: $\{1, 1/2, 1/3, 1/4, 1/5\}$

Series: $\tan x = x + x^3/3 + 2x^5/15 + 17x^7/315 + \ldots$

Index: $\{i: i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Coeff: $\{a_i: i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Sign: $\{+1: i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Mgtde: $\{1, 1/3, 2/15, 17/315, 62/2835\}$

Series: $\sec x = 1 + x^2/2 + 5x^4/24 + 61x^6/720 + \ldots$

Index: $\{i: i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

Coeff: $\{a_i: i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

Sign: $\{+1: i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

Mgtde: $\{1, 1/2, 5/24, 61/720, 277/8064\}$

Series: $\arcsin x = x + 1/2 * x^3/3 + 1/2 * 3/4 * x^5/5 + 1/2 * 3/4 * 5/6 * x^7/7 + \ldots$

Index: $\{i: i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Coeff: $\{a_i: i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Sign: $\{+1: i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Mgtde: $\{1, 1/6, 3/40, 5/112, 35/1152\}$

Series: $\arctan x = x - x^3/3 + x^5/5 - x^7/7 + \ldots$

Index: $\{i\colon i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Coeff: $\{a_i\colon i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Sign: $\{+1\colon i \text{ Qmod } 4 = 1 \wedge i \geq 0\} \vee \{-1\colon i \text{ Qmod } 4 = 3 \wedge i \geq 0\}$

Mgtde: $\{1, {}^1/_3, {}^1/_5, {}^1/_7, {}^1/_9\}$

Series: $\sinh x = x + x^3/3! + x^5/5! + x^7/7! + \ldots$

Index: $\{i\colon i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Coeff: $\{a_i\colon i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Sign: $\{+1\colon i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Mgtde: $\{1, {}^1/_6, {}^1/_{120}, {}^1/_{5040}, {}^1/_{362880}\}$

Series: $\cosh x = 1 + x^2/2! + x^4/4! + x^6/6! + \ldots$

Index: $\{i\colon i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

Coeff: $\{a_i\colon i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

Sign: $\{+1\colon i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

Mgtde: $\{1, {}^1/_2, {}^1/_{24}, {}^1/_{720}, {}^1/_{40320}\}$

Series: $\tanh x = 1 - x^3/3 + 2x^5/15 - 17x^7/315 + \ldots$

Index: $\{i\colon i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Coeff: $\{a_i\colon i \text{ Qmod } 2 = 1 \wedge i \geq 0\}$

Sign: $\{+1\colon i \text{ Qmod } 4 = 1 \wedge i \geq 0\} \vee \{-1\colon i \text{ Qmod } 4 = 3 \wedge i \geq 0\}$

Mgtde: $\{1, {}^1/_3, {}^2/_{15}, {}^{17}/_{315}, {}^{62}/_{2835}\}$

Series: $\operatorname{sech} x = 1 - x^2/2 + 5x^4/24 - 61x^6/720 + \ldots$

Index: $\{i\colon i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

Coeff: $\{a_i\colon i \text{ Qmod } 2 = 0 \wedge i \geq 0\}$

Sign: $\{+1\colon i \text{ Qmod } 4 = 0 \wedge i \geq 0\} \vee \{-1\colon i \text{ Qmod } 4 = 2 \wedge i \geq 0\}$

Mgtde: $\{1, {}^1/_2, {}^5/_{25}, {}^{61}/_{720}, {}^{277}/_{8064}\}$

Series: $\operatorname{arcsinh} x = x - {}^1/_2{*}x^3/3 + {}^1/_2{*}{}^3/_4{*}x^5/5 - {}^1/_2{*}{}^3/_4{*}{}^5/_6{*}x^7/7 + \ldots$

Index: $\{i\colon i \operatorname{Qmod} 2 = 1 \wedge i \geq 0\}$

Coeff: $\{a_i\colon i \operatorname{Qmod} 2 = 1 \wedge i \geq 0\}$

Sign: $\{+1\colon i \operatorname{Qmod} 4 = 1 \wedge i \geq 0\} \vee \{-1\colon i \operatorname{Qmod} 4 = 3 \wedge i \geq 0\}$

Mgtde: $\{1, {}^1/_6, {}^3/_{40}, {}^5/_{112}, {}^{35}/_{1152}\}$

Series: $\operatorname{arctanh} x = x + x^3/3 + x^5/5 + x^7/7 + \ldots$

Index: $\{i\colon i \operatorname{Qmod} 2 = 1 \wedge i \geq 0\}$

Coeff: $\{a_i\colon i \operatorname{Qmod} 2 = 1 \wedge i \geq 0\}$

Sign: $\{+1\colon i \operatorname{Qmod} 2 = 1 \wedge i \geq 0\}$

Mgtde: $\{1, {}^1/_3, {}^1/_5, {}^1/_7, {}^1/_9\}$

Bibliography

Abbreviations:

AAAI	American Association for Artificial Intelligence (*syn.* National Conference on Artificial Intelligence)
ACM	Association for Computing Machinery
AISB	Artificial Intelligence and the Simulation of Behaviour
ECAI	European Conference on Artificial Intelligence
IEEE	Institute of Electrical and Electronics Engineers
IJCAI	International Joint Conference on Artificial Intelligence
MIT	Massachusetts Institute of Technology
SIGART	Special Interest Group on Artificial Intelligence

[Acton & Squire 85] R. A. Acton & P. T. Squire, *Solving Equations with Physical Understanding*, Adam Hilger Ltd., Bristol (1985)

[Allen 83] J. Allen, *Maintaining Knowledge about Temporal Intervals*, Communication of the ACM, vol. 26, no. 11, pp. 832-843 (1983)

[Ames 77] W. F. Ames, *Numerical Methods for Partial Differential Equations*, 2nd ed., Academic Press, (1977)

[Baker 84] A. Baker, *A Concise Introduction to the Theory of Numbers*, Cambridge University Press (1984)

[Bennett 87] S. Bennett, *Approximation in Mathematical Domains*, Proc. IJCAI-87, Morgan Kaufmann (1987)

[Berleant & Kuipers 88] D. Berleant & B. Kuipers, *Using Incomplete Quantitative Knowledge in Qualitative Reasoning*, Proc. AAAI-88, Morgan Kaufmann (1988)

[Boas 66] M. L. Boas, *Mathematical Methods in the Physical Sciences*, New York, Wiley (1966)

[Bobrow 84] D. Bobrow (ed.), *Special Volume on Qualitative Reasoning about Physical Systems*, Artificial Intelligence, 24 (1984)

[Braun 78] M. Braun, *Differential Equations and their Applications*, Applied Mathematical Sciences 15, 2nd ed., Springer Verlag (1978)

[de Bruijn 58] N. G. de Bruijn, *Asymptotic Methods in Analysis*, North-Holland, Amsterdam (1958)

[Bundy 83] A. Bundy, *The Computer Modelling of Mathematical Reasoning*, Academic Press (1983)

[Bundy & Welham 81] A. Bundy & B. Welham, *Using Meta-level Inference for the Selective Application of Multiple Rewrite Rules in Algebraic Manipulation*, Artificial Intelligence, vol. 16, no. 2 (1981)

[Bundy & Silver 81] A. Bundy & B. Silver, *Homogenization: Preparing Equations for Change of Unknown*, Department of Artificial Intelligence, Research Paper, 159, University of Edinburgh (1981)

[Bundy & Sterling 81] A. Bundy & L. Sterling, *Meta-level Inference in Algebra*, Department of Artificial Intelligence, Research Paper, 164, University of Edinburgh (1981)

[Char *et al.* 85] B. Char *et al.*, *A Tutorial Introduction to MAPLE*, Research Paper CS-85-56, Univeristy of Waterloo, Canada (1985)

[Coddington 62] E. A. Coddington, *An Introduction to Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, New Jersey (1962)

[Cohn & Hovsepian 88] A. Cohn & F. Hovsepian, *Towards a Calculus of Quality Spaces & Measuring Scales Based on Tolerance Relations*, Department of Computer Science, University of Warwick (1988)

[Copson 79] E. T. Copson, *Metric Spaces*, Cambridge Tracts in Mathematics, 57, Cambridge University Press (1979)

[Crandall 56] S. H. Crandall, *Engineering Analysis*, Mc Graw-Hill Book Company, Inc. (1956)

[Dague *et al.* 87] P. Dague, O. Raiman & P. Deves, *Troubleshooting when modelling is the trouble*, Proc. AAAI-87, Morgan Kaufmann (1987)

[Davis 87a] E. Davis, *Order of Magnitude Reasoning in Qualitative Differential Equations*, Technical Report 312, New York University (1987)

[Davis 87b] E. Davis, *Constraint Propagation with Interval Labels*, Artificial Intelligence, 32, pp281-331 (1987)

[Davis & Hersch 72] M. Davis & R. Hersch, *Nonstandard Analysis*, Scientific American, vol. 226, no. 6, pp.78-86, June (1972)

[Dixon & de Kleer 88] M. Dixon & J. de Kleer, *Massively Parallel Assumption-Based Truth Maintenance*, Proc. AAAI-88, Morgan Kaufmann (1988)

[Dormoy 88] J. Dormoy, *Controlling Qualitative Resolution*, Proc. AAAI-88, Morgan Kaufmann (1988)

[Dormoy & Raiman 88] J. Dormoy & O. Raiman, *Assembling A Device*, Proc. AAAI-88, Morgan Kaufmann (1988)

[Ehrig 79] H. Ehrig, *Introduction to the Algebraic Theory of Graph Grammars (A Survey)*, in Graph Grammars and their Application to Computer Science and Biology, (eds. Claus, V., Ehrig, H., and Rozenberg, G.) Lecture Notes in Computer Science, Springer-Verlag (1979)

[Finlayson 72] B. Finlayson, *The Method of Weighted Residuals and Variational Principles*, Mathematics in Science & Engineering Series, vol. 87, Academic Press (1972)

[Forbus 84] K. Forbus, *Qualitative Process Theory*, Artificial Intelligence, 24, pp. 85-168 (1984)

[Hayes 79] P. Hayes, *The Naive Physics Manifesto*, in Expert Systems in the Micro Electronic Age, ed. D. Michie, Edinburgh University Press (1979)

[Hayes 85] P. Hayes, *The Second Naive Physics Manifesto*, in *Formal Theories of the Commonsense World*, ed. J. Hobbs & R. Moore, Ablex, Norwood, New Jersey (1985)

[Hobbs & Moore 85] J. Hobbs & R. Moore, *Formal Theories of the Commonsense World*, Ablex Series in Artificial Intelligence, Ablex Publishing Corporation, New Jersey (1985)

[Hogge 87] J. Hogge, *Compiling Plan Operators from Domains Expressed in Qualitative Process Theory*, Proc. AAAI-87, Morgan Kaufmann (1987)

[Iwasaki 88] Y. Iwasaki, *Causal Ordering in a Mixed Structure*, Proc. AAAI-88, Morgan Kaufmann (1988)

[Iwasaki & Bhandari 88] Y. Iwasaki & Bhandari, *Formal Basis for Commonsense Abstraction of Dynamic Systems*, Proc. AAAI-88, Morgan Kaufmann (1988)

[Iwasaki & Simon 86a] Y. Iwasaki & H. Simon, *Causality in Device Behaviour*, Artificial Intelligence, 29 pp3-32 (1986a)

[Iwasaki & Simon 86b] Y. Iwasaki & H. Simon, *Theories of Causal Ordering: reply to de Kleer and Brown*, Artificial Intelligence, 29 pp63-72 (1986b)

[Jeffreys 62] H. Jeffreys, *Asymptotic Approximations*, Oxford, Clarendon Press (1962)

[Jeffreys & Jeffreys 56] H. Jeffreys & B. Jeffreys, *Methods of Mathematical Physics*, 3rd ed., Cambridge University Press (1956)

[de Kleer 75] J. de Kleer, *Qualitative & Quantitative Knowledge in Classical Mechanics*, AI-TR-352, M.I.T. Artificial Intelligence Laboratory (1975)

[de Kleer 77] J. de Kleer, *Multiple Representations of Knowledge in a Mechanics Problem Solver*, Proc. IJCAI-77 (1977)

[de Kleer 86] J. de Kleer, *An Assumption Based Truth Maintenance System*, Artificial Intelligence, 28 (1986)

[de Kleer & Brown 84] J. de Kleer & J. Brown, *A Qualitative Physics Based on Confluences*, Artificial Intelligence, 24 pp7-83 (1984)

[Kreider *et al.* 80] D. Kreider *et al.*, *An Introduction to Linear Analysis*, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, U.S.A. (1980)

[Kuipers 85] B. Kuipers, *The Limits of Qualitative Simulation*, IJCAI-85, pp. 128-136, Morgan Kaufmann (1985)

[Kuipers 86a] B. Kuipers, *Qualitative Simulation*, Artificial Intelligence, 29 pp289-338 (1986a)

[Kuipers 86b] B. Kuipers, *Qualitative Simulation as Causal Explanation*, AI TR86-24, University of Texas at Austin (1986b)

[Kuipers 86c] B. Kuipers, *Building a Process Model of Evaporation*, (incomplete and unpublished) (1986)

[Kuipers 87] B. Kuipers, *Abstraction by Time-Scale in Qualitative Simulation*, Proc. AAAI-87, pp. 621-625, Morgan Kaufmann (1987)

[Kuipers 88] B. Kuipers, *The Qualitative Calculus is Sound but Incomplete: A Reply to Peter Struss*, Artificial Intelligence in Engineering, vol. 3, no. 3 (1988)

[Kuipers & Chiu 87] B. Kuipers & C. Chiu, *Taming Intractible Branching in Qualitative Simulation*, Proc. IJCAI-87, pp. 1079-1085, Morgan Kaufmann (1987)

[Kuipers & Kassirer 83] B. Kuipers & J. P. Kassirer, *How to Discover a Knowledge Representation for Causal Reasoning by Studying an Expert Physician*, Proc. IJCAI-83, Morgan Kaufmann (1983)

[Landshoff & Metherell 79] P. Landshoff, & A. Metherell, , *Simple Quantum Physics*, Cambridge University Press (1979)

[Lee & Kuipers, 88] W. Lee & B. Kuipers, *Non-intersection of Trajectories in Qualitative Phase Space: A Global Constraint for Qualitative Simulation*, Proc. AAAI-88, Morgan Kaufmann (1988)

[Lefschetz 77] S. Lefschetz, *Differential Equations: Geometric Theory*, Dover Publications Inc., New York (1977)

[Lutz 86] R. Lutz, *Diagram Parsing - A New Technique for Artificial Intelligence*, Cognitive Studies Research Papers, CSRP.054, University of Sussex (1986)

[Martin & Fateman 71] W. A. Martin & R. J. Fateman, *The MACSYMA System*, in S. R. Petrick (ed.), Proc. 2nd Symposium on Symbolic & Algebraic Manipulation, pp.59-75, ACM (1971)

[Mathews & Walker 70] J. Matthews & R. L. Walker, *Mathematical Methods of Physics*, 2nd. ed. W. A. Benjamin, Inc., New York (1970)

[Mavrovouniotis 89] M. Mavrovouniotis, *Constraint Propagation with N-ary Semiquantitative Relations*, Proc. AISB, Brighton, Sussex (1989)

[Mavrovouniotis & Stephanopoulos 87] M. Mavrovouniotis & G. Stephanopoulos, *Reasoning with Orders of Magnitude and Approximate Relations*, Proc. AAAI-87, pp. 626-631, Morgan Kaufmann (1987)

[Morgan 87] A. J. Morgan, *Predicting the Behaviour of Dynamic Systems Using Qualitative Vectors*, in *Advances in Artificial Intelligence*, (eds. J. Hallam & C. Mellish), AISB, pp.81-95 (1987)

[Morgan 88] A. J. Morgan, *The Qualitative Behaviour of Physical Systems*, Ph.D. thesis, University of Cambridge (1988)

[Pfaltz & Rosenfeld 69] J. Pfaltz & A. Rosenfeld, *Web Grammars*, Proc. IJCAI-69 pp. 609-619 (1969)

[Pipes & Harvill 70] L. Pipes & L. Harvill, *Applied Mathematics for Engineers and Physicists*, 3rd ed., McGraw-Hill (1970)

[Raiman 86] O. Raiman, *Order of Magnitude Reasoning*, Proc. AAAI-86, pp. 105-112, Morgan Kaufmann (1986)

[Rich 81] C. Rich, *Inspection Methods in Programming*, M.I.T. Artificial Intelligence Laboratory, AI-TR-604 (1981)

[Robinson 66] A. Robinson, *Non-Standard Analysis*, North-Holland Publishing Company, Amsterdam (1966)

[Rosenfeld & Milgram 72] A. Rosenfeld & D. Milgram, "Web Automata and Web Grammars", Machine Intelligence, 7 pp 307-324 (eds. B. Meltzer & D. Michie), Edinburgh University Press (1972)

[Sacks 85a] E. Sacks, *Qualitative Mathematical Reasoning*, Proc. IJCAI-85, Morgan Kaufmann (1985)

[Sacks 85b] E. Sacks, *Qualitative Mathematical Reasoning*, LCS/TR-329, M.I.T. (1985)

[Sacks 87a] E. Sacks, *Qualitative Sketching of Parameterized Functions*, In D. Sriram and R.A. Adey, editors, *Knowledge Based Expert Systems for Engineering: Classification, Education and Control*, pp1-13, Computational Mechanics Publications, Boston (1987)

[Sacks 87b] E. Sacks, *Piecewise Linear Reasoning*, Proc. AAAI-87, Morgan Kaufmann (1987)

[Sacks 87c] E. Sacks, *Hierarchical Reasoning about Inequalities*, Proc. AAAI-87, Morgan Kaufmann (1987)

[Sacks 88] E. Sacks, *Automatic Qualitative Analysis of Ordinary Differential Equations Using Piecewise Linear Approximations*, M.I.T./LCS/TR–416, Massachusetts Institute of Technology (1988)

[Simmons 86] R. Simmons, "*Commonsense*" *Arithmetic Reasoning*, Proc. AAAI-87, Morgan Kaufmann (1987)

[Spiegel 68] M. R. Spiegel, *Mathematical Handbook*, Schaum's Outline Series in Mathematics, McGraw-Hill Book Company (1968)

[Spiegel 74] M. R. Spiegel, *Advanced Calculus*, Schaum's Outline Series in Mathematics, McGraw-Hill Book Company (1974)

[Steel & de Roeck 87]. S. Steel & de Roeck, *Bidirectional Chart Parsing*, in *Advances in Artificial Intelligence* (eds. J.Hallam & C.Mellish), AISB, John Wiley & Sons pp223-235 (1987)

[Stephenson 78] G. Stephenson, *Mathematical Methods for Science Students*, 2nd *ed.*, Longman, London (1978)

[Sterling *et al.* 82] L. Sterling *et al.*, *Solving Symbolic Equations with PRESS*, Department of Artificial Intelligence, Research Paper 171, University of Edinburgh (1982)

[Stevens & Gentner 83] A. Stevens & D. Gentner, *Mental Models*, Lawrence Earlbaum, New Jersey (1983)

[Struss 87] P. Struss, *Problems with Interval Based Qualitative Reasoning*, ZTI INF 2, Siemens, Munich (1987)

[Struss 88a] P. Struss, *Mathematical Aspects of Qualitative Reasoning*, ZTI INF 31, Siemens, Munich (1988a)

[Struss 88b] P. Struss, *Mathematical Aspects of Qualitative Reasoning, Part Two, Differential Equations*, ZTI INF 22, Siemens, Munich (1988b)

[Struss 88c] P. Struss, *Mathematical Aspects of Qualitative Reasoning*, Artificial Intelligence in Engineering, vol. 3, no. 3 (1988)

[Struss 88d] P. Struss, *Extensions to ATMS-Based Diagnosis*, Proc. 3rd International Conference on Applications of Artificial Intelligence in Engineering, Computational Mechanics Publications (1988)

[Struss 88e] P. Struss, *Global Filters for Qualitative Behaviours*, Proc. AAAI-88, Morgan Kaufmann (1988)

[Weld 85] D. Weld, *Combining Discrete and Continuous Process Models*, Proc. IJCAI-85 Morgan Kaufmann (1985)

[Weld 86] D. Weld, *The Use of Aggregation in Causal Simulation*, Artificial Intelligence, 30 pp1-34 (1986)

[Weld 87] D. Weld, *Comparative Analysis*, Proc. IJCAI-87, pp. 959-965, Morgan Kaufmann (1987)

[Weld 88a] D. Weld, *Exaggeration*, Proc. AAAI-88, Morgan Kaufmann (1988)

[Weld 88b] D. Weld, *Choices for Comparative Analysis: DQ analysis or exaggeration?*, Artificial Intelligence in Engineering, vol. 3, no. 3 (1988)

[Weld 88c] D. Weld, *Comparative Analysis*, Artificial Intelligence, 36 pp333-373 (1988)

[Weld 88d] D. Weld, Theories of Comparative Analysis, Ph.D. thesis, AI-TR 1035, Artificial Intelligence Laboratory, M.I.T. (1988)

[Wenger 87] E. Wenger, *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann Publishers Inc., Los Altos, California (1987)

[Williams 86] B. Williams, *Doing Time: Putting Qualitative Reasoning on Firmer Ground*, Proc. AAAI-86, Morgan Kaufmann (1986)

[Williams 87] B. Williams, *Principled Design Based on Qualitative Behavioural Descriptions*, proposal for doctoral research, M.I.T. Artificial Intelligence Laboratory (1987)

[Williams 88] B. Williams, *MINIMA: A Symbolic Approach to Qualitative Algebraic Reasoning*, Proc. AAAI-88, Morgan Kaufmann (1988)

[Williams 88] C. P. Williams, *Analytic Abduction from QSIM*, Research Paper 373, Department of Artificial Intelligence, University of Edinburgh (1988)

[Yip 87] K. Yip, *Extracting Qualitative Dynamics from Numerical Experiments*, Proc. AAAI-87, Morgan Kaufmann (1987)

[Yip 88] K. Yip, *Generating Global Behaviours Using Deep Knowledge of Local Dynamics*, Proc. AAAI-88, Morgan Kaufmann (1988)